Development of a Novel Distributed Wearable Sensor Platform

by

Tarek Nasser El Harake

BEng, Carleton University

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree

of

Master of Applied Science in Biomedical Engineering

Department of Systems and Computer Engineering

Carleton University, Ottawa, Ontario, Canada, K1S 5B6

Abstract

Recent advancements in low-power, low-cost, miniaturized sensor technology have propelled wearable electronics into the mainstream. Wearable devices like smartwatches, smartglasses and other smart clothing are now commonplace, finding applications in health and fitness, sports analytics, personnel tracking, and human computer interfaces. Many of these systems, however, are consumer oriented, with closed environments and limited access to sensor data, greatly restricting their use as exploratory tools for experimenters and researchers. Some research devices exist but are often expensive and limited in what they offer. In this work, we describe the development of a novel wearable sensor platform targeted towards researchers, developers, and hobbyists, that attempts to overcome many of the limitations of existing systems while adding new features to improve and simplify experimental designs. Following a clear set of guidelines, the system was built, validated, and tested in a real-life experiment to assess its effectiveness and ease of use. The developed platform consists of small 35x25x15mm nodes each containing an nRF52 Bluetooth microcontroller, IMU sensor, and small LiPo battery. Two additional input and output nodes allow external devices and sensor connections. The nodes communicate wirelessly with a central hub using Bluetooth 5, sending raw sensor data, which is then visualized and analyzed using a graphical software interface.

Acknowledgements

I would like to thank my supervisor Dr. Andy Adler for his continuous help and guidance over the past few years. His support and encouragement have been instrumental in the completion of this work.

I would also like to thank my family for all the support they've provided me throughout this endeavour.

Contents

List of Figures

List of Abbreviations

ADC Analog to Digital Converter

API Application Programming Interface

CAD Computer Aided Design

COM Communication Port

CSV Comma Separated Values

ECG Electrocardiogram

EEG Electroencephalogram

EMG Electromyogram

GSR Galvanic Skin Response

HID Human Interface Device

I/O Inputs/Outputs

I2C Inter-Integrated Circuit

IC Integrated Circuit

IDE Integrated Development Environment

IMU Inertial Measurement Unit

KST An open source visualization software

LDO Linear Drop Out

LED Light Emitting Diode

LiPo Lithium Polymer

OBS Open Broadcast Software

PCB Printed Circuit Board

PPG Photoplethysmography

RTOS Real-Time Operating System

SWD Serial Wire Debug

TTL Transistor-Transistor Logic

TXT Text file format

UART Universal Asynchronous Receiver/Transmitter

USB Universal Serial Bus

Chapter 1

Introduction

1.1 Advances in wearable technology

Over the past decade, the rapid innovation in sensor technologies coupled with the ever-growing needs of the smartphone market has resulted in the development of miniaturized, low cost, low power, embedded sensors for a variety of uses. For example, a single smartphone, the Samsung Galaxy S10 released in 2019, sports an entire sensor suite, from motion and light sensors, to an ultrasonic fingerprint scanner and a heart rate monitor, in addition to the more traditional camera and microphone sensors [1].

The availability of such sensors has birthed entire new technology sectors such as the Internet of Things or IoT, which relies on wireless embedded sensor networks to digitize and connect the virtual world with the real one. Sensors play a large role in enabling IoT as it allows such devices to record many real world parameters to be later used for analysis [2]. Home automation is a subset of IoT which uses a sensor network deployed in a home to automatically control various aspects of the house based on sensor data analysis. For example, passive infrared sensors are often used to detect room occupancy and control lighting accordingly, smoke and gas detectors can detect gas leaks, while

security oriented devices such as glass break sensors and motion sensors can detect intruders [3].

Another subset of IoT that has found great success in the past few years is wearable technology or simply "wearables", which benefit from the continuation of sensor miniaturization and accessibility. Wearables are electronic devices that are small enough to be worn on the human body or integrated into regular clothing items [4][5][6][7]. These systems use embedded sensors to collect data from the wearer and extract useful information related to the person's fitness, health, position, environment and many other parameters, opening such systems to be used in a variety of applications and a number of emergent fields. The first wearable product to find commercial success was the Fitbit released around 2009, this rubber band used an embedded inertial measurement unit to measure its wearer's motion and estimate their step count and energy expenditure [5]. Over the following years, many consumer electronic companies released their own wearable devices from fitness trackers and smart watches such as the Samsung Gear (2013) [8] and Apple Watch (2014) [6], to smart glasses such as the Google Glass (2013) [9], and Virtual/Augmented Reality headsets like the Microsoft HoloLens (2016) [10].

This explosion in wearable sensor technologies offers new insights that was previously inaccessible in many applications and fields related to human monitoring and human computer interaction. With such sensors, it is now possible for the first time to measure human experiences continuously, completely wirelessly and without impeding the wearer's activity [4][7].

1.2 Applications of wearable sensors

One of the largest applications of wearable technology is fitness and healthcare [11][7]. The nature of wearables allows the use of sensors in close proximity to the human body

and sometimes in contact with the skin. Such placements are ideal for health-related sensors that can measure a wide variety of biological signals such as the motion of the body, skin temperature measurements, skin oxygen saturation, and electrical biosignals such as ECG for cardiovascular information, EMG for muscle activity, and EEG for brain activity among many others. These measurements can then be used to extract important information about the wearer's health and fitness levels. Most modern smartwatches contain a number of these sensors and provide the user software interfaces to monitor and keep track of their health and fitness over time [5][6][8].

In addition to consumer oriented applications, the miniaturization of clinical grade biosignal sensors has allowed health monitoring not only for in- and out-patients, but also for the general public with preemptive telehealth monitoring (and eventually care) [12]. One such example is the Apple Heart Study [13] developed in partnership with Stanford University to detect and report atrial fibrillation and other heart conditions in Apple watch wearers.

Many of these same sensors have made their way outside of the healthcare domain finding uses in professional sports training. Coaches and athletes are looking to monitor their training and characterize and evaluate improvements. Multiple companies such as Polar and Catapult offer solutions for hockey, soccer, and other team sports that use IMUs, GPS receivers, and heart rate sensors to provide a network of sensors for real-time player tracking and sports analytics [14][7].

Human Computer Interfaces are another prime application for wearable technologies. Modern Virtual and Augmented Reality systems have head mounted displays with inward and outward facing sensors and controllers, which use a combination of IMUs, cameras, infrared receivers, and capacitive touch sensors among others, capturing the user's body movements, hand gestures, and even face and eye motion, to build immersive gaming experiences as well as telepresence and telecontrol applications [15][7].

A number of emergent fields have also found great benefit from the use of wearable technologies. From physiological and situational awareness systems for military and search and rescue operations [16], to keeping track of miner's health, condition, and location during mining operations [17][18], and even tracking the health and location of cows for the cattle industry [19].

1.3 Wearable sensor evolution

As new sensor modalities and form factors continue to emerge, the wearables field is bound to keep expanding. And as with many new technologies, much still needs to be discovered in regards to the potential of wearable technologies, their applications, and role in society. Current wearable systems are mostly focused on end-user applications, with most of the research and development for these product conduced behind closed doors. As such, the knowledge of wearable and sensor development is not often shared between developers, while public researchers and hobbyists are constrained to design and build their own systems from scratch.

In recent years, some researchers have leveraged these advancements in technology to develop new sensor platforms with specific applications in mind [20].

The Nightingale V2 [21] (developed at IMEC Netherlands) is a Bluetooth multi-sensor platform incorporating a number of bio-signal sensors including ECG, Bio-impedance, PPG, motion and heart sounds sensors, and targeted towards synchronous multi-parameter sensing in health monitoring applications.

NETWIS [22](developed at the University of Bologna and ETH Zurich) is a wireless sensor network consisting of multiple small motion sensing Wireless Inertial Sensor (WIS) devices which communicate wirelessly to a gateway using custom protocols, promising high scalability and synchronization for full body motion detection applica-

tions.

BlueSense [23] (developed at the University of Sussex) is a real-time, low power, extensible wearable platform targeted towards full body motion sensing. It is based around an ATmega microcontroller and contains an Inertial Measurement Unit and various methods of data logging including an on-board SD card, Bluetooth and USB interfaces.

However, for many others, designing and building a sensor platform from the ground up is often times outside the scope of the research.

During this technology's growth stage, it is vital for developers to experiment and test new ideas for novel transducer designs, form factors, body locations, wireless protocols, and sensor data analysis. This can be clearly seen in the design of commercial wearable devices such as Photoplethysmography (PPG) wristbands [24], which are constantly and rapidly evolving with each iteration of the product to accommodate the highly dynamic environment that is the human body.

It becomes thus clear that there is a need for a solution offering researchers and developers in various fields an environment to rapidly experiment and enact changes, with support for a wide range of sensors, rapid changes in sensor design and open data for in-depth analysis. While most consumer products remain largely closed to experimenters, with only barebone APIs and offering little control [25][6], a small number of companies have released wearable and sensor products with a focus towards the research market and experimental design.

1.4 Review of current systems

Many wearable devices currently exist on the market from consumer oriented products to research focused devices. In this section we review five popular system and summarize our findings in Table 1.1.

Perhaps the most recognizable wearable currently on the market is the **Apple Watch**. This smartwatch, introduced by Apple in 2015, offers a wide range of features in a compact and sleek design targeted towards the general public. The new generations of the device have included a suite of sensors from motion and light detectors to optical (PPG) as well as electrical (ECG) heart monitors [6]. This has allowed the watch to offer additional features to its users from fitness applications that measures and compares heart rate levels, to full on medical uses by collecting ECG information, all without the need for external devices such as smartphones or computers for analysis. Although useful for many applications, the quality levels of these sensors, as well as the inability to extract the raw data from the device makes it less than ideal for most research studies.

The Finnish company **Polar** has been in the wearables market for the past four decades with their release of the first wearable heart rate monitor in 1982 [26]. Today, they continue to produce smartwatches with optical heart rate monitors, but are perhaps more famous for their high accuracy ECG chest straps for fitness and training applications [25]. However, similarly to the Apple Watch, these devices are targeted towards the consumer market, and provides little in terms of access to the raw data, opting instead to send simple heart rate measurements every few seconds to the user's mobile device.

The **Empatica E4** wristband moves away from the consumer market and promises clinical quality data to researchers interested in capturing real-time wrist based phys-

iological data. It features sensors for measuring heart rate, motion, temperature, as well as Galvanic Skin Response (GSR), in a simple wristband form factor [27]. Unlike its consumer counterparts, the E4 allows researchers to access its raw data, for more in-depth analysis. However, it's research-oriented nature comes at a fairly high price markup compared to similarly capable devices, making it inaccessible for many users.

The **BioRadio** is another wearable research device focused mostly on biomedical measurements. With internal motion sensors, and four biopotential channels, the BioRadio is ideal for taking high resolution measurements of electrical parameters such as ECG, EMG, and EEG [28]. The system also supports additional plug-in sensors such as respiration and blood pressure through a custom interface. Although technically wearable, the BioRadio is large and heavy compared to other systems with a form factor of a small phone and meant to be clipped on the waist. This, in addition to its long sensor wires can restrict the motion of its wearer making it cumbersome for many applications requiring some level of dexterity.

The **Shimmer3** is a wearable sensor platform targeted towards researchers and academics offering a selection of small units with different sensors, including IMU, ECG, EMG and GSR sensor units [29]. A software solution called Consensys allows the researcher to then collect data from the various units with the ability to stream from up to 7 devices concurrently. The devices are reasonably sized at 65 x 32 x12 mm, allowing them to be strapped to various limbs without much impact on the wearer. However, similar to the previous research devices mentioned, the price of the Shimmer3 system makes it prohibitive to many hobbyists and small scale researchers.

System	**Apple Watch** [6]	**Polar Strap** [25]	**BioRadio** [28]	**Empatica E4** [27]	**Shimmer3** [29]	**This Work**
Application	Health/Fitness	Fitness/Sports	Research	Research	Research/Dev	Hobby/Research/Dev
Multisensor Network	No	No	No	No	Optional (up to 7)	5 Nodes
Modularity	No	No	Limited (External Sensors)	No	Limited (Custom Board)	Yes (Plug and Play)
Output Feedback	No	No	No	No	No	Yes (Output Node)
Data Access	Limited	Limited	Open	Open	Open	Open
External Sync	No	No	No	No	Yes (External Sensors)	Yes (2 Video Inputs)
Size	Small (40x34x10.7mm)	Medium (65x34x10mm)	Large (100x60x20mm)	Small (44x40x16mm)	Small (51x34x14mm)	Very Small (35x25x15mm)
Cost	$$	$	$$$	$$$	$$$$	$$

Table 1.1: Wearable systems comparison

1.5 Limitations of current systems

As diverse as these systems are, they all possess a number of limitations that hinder their use in many experiments and applications. These limitations, related to the form factor, sensor selection, and other design decisions, severely limits the flexibility of these systems and restricts their use to a select number of situations.

Monolithic: Most of theses systems have a monolithic design with all the sensors enclosed in one case. This simplifies the complexities of the system for both the manufacturer and the user. However, such a design is only capable of measuring sensor information from one location on the body, or requires a number of external wired connections to reach the desired sensing areas. A monolithic design disqualifies any experiment that requires collecting various parameters from distant parts of the body, or at the very least inconveniences the user by requiring cumbersome wired connections.

Form factor: Size and form factor are highly important in wearable systems as this directly dictates the location at which the device could be placed as well as the type of activities possible while wearing the device. The smaller a wearable device is, the more versatile it becomes, allowing it to be placed on the body without impeding the wearer's motion or affecting their comfort. This is particularly important for continuous applications like monitoring sleep or when the wearer needs to retain full range of motion for training and fitness applications. Miniaturization comes with a number of inherent drawbacks like small battery life and relatively low range, however for many applications and experiments, a small form factor is an essential prerequisite.

Sensor selection: The wearable system's sensor selection often defines its uses and the applications it might be suitable for. Modern systems incorporate multiple sensors that can record multi-parameter data concurrently. However, sensor development and prototyping can sometimes be just as vital to an experimental setting as the development

of the entire system. Unfortunately, many of these systems do not allow that level of customization, and are incapable of interfacing to the developer's custom sensor solution, reducing the amount of control over the types of experiment that can be conducted. Developers are thus limited to the sensors offered by the manufacturer and are incapable of making modifications to the sensor's internal configurations.

Limited data: An issue more prevalent in consumer oriented devices is the limited access to the raw data collected by the sensors. The data collected by consumer wearables is often made hard or impossible to access by the user, who might only be given access to a heavily processed version, giving them a simple high level overview. A common example can be seen in heart rate chest bands, which although collects rich ECG data, only exposes a simple Heart Rate number every few seconds. The severity of this problem varies between devices: while some do not expose any data at all, opting to send it directly to the cloud for analysis, others offer limited APIs to collect simple processed data, and more research oriented systems allow users to collect the raw data or lightly processed versions of it.

Cost: While consumer devices are lacking when it comes to sensor selection and access to data, it has a large advantage over research devices when it comes to cost. Many research wearables cost thousands of dollars, which excludes small research groups and hobbyists from the list of potential users. This is a major oversight of current systems since such groups are instrumental in the exploration of new ideas and the development of innovative systems.

Features: There is also a distinct lack of features that would seem to be important from an experimental point of view, such as a feedback system that would not only allow real-time data collection but also real-time feedback through actuator devices such as motors, lights, sound, and haptic feedback devices. Or a way to synchronize data from external devices such as video cameras in order to overlap sensor data with video footage,

providing even more insight into the conducted experiments. Such ideas with a focus on experimental design and device development could thus provide an additional layer of integration to custom and third party devices.

1.6 Purpose

The purpose of this research is to build a wearable sensor platform that overcomes the limitations of many of the devices currently on the market and offers a new alternative for hobbyists, researchers, and developers looking to rapidly setup experiments and explore data from a variety of traditional and custom sensors. The system should not be application-, sensor-, or location-specific, focusing instead on a distributed, modular approach, giving the user control over the sensor modality, position, and configuration. Additionally, as one of the most crucial aspects of wearables, the system should be designed to be as small and lightweight as possible, reducing the effects on the wearer's comfort and range of motion. Finally, to appeal to a broader audience, we will design the system to be easy to use and configure, with a simple and intuitive user interface that allows users to easily setup and record new experiments.

Chapter 2

Thesis Overview

2.1 Guiding Principles

- **Distributed:** to measure multiple independent parameters simultaneously.
- **Modular:** to allow various configurations and versatile positioning on the body.
- **Extendable:** to allow the use of many sensor modules including custom designs.
- **Small and lightweight:** to be placed on the body without interfering with the wearer's range of motion or comfort levels.
- **Easy to setup and modify on the spot:** ideal for rapid experimentation and exploration.
- **Raw sensor data:** with timestamps and headers, and a simple file storage for streamlined analysis.
- **Low latency for real-time communication:** to allow the potential for feedback loops and detect events as they occur.
- **Simple and intuitive:** graphical user interfaces and user experience with simple one button operation.

2.2 System Description

Figure 2.1: Overview of the system, including the nodes, hub, and PC application.

Based on those guiding principles we have built a complete distributed sensor platform, consisting of wireless sensor modules called nodes, a central network router called the hub, and complemented by a comprehensive computer application with a number of advanced software features. Each of these three subsystems play a central role in the operation of the platform.

The nodes are small and lightweight wearable devices, each containing a microcontroller, Bluetooth radio, small lithium polymer battery, as well as a sensor, and act as the source for data collection. The nodes collect data independently from one another and sends it in real-time to the central hub through a real-time, high-throughput Bluetooth transmission link.

The hub's main role is to receive and buffer the data sent by the nodes, creating a sensor network with a star topology. Other than controlling the network by connecting to, synchronizing, and communicating with the nodes, the hub also acts as an interme-

diary between the sensor network and computer software operated by the user through a USB interface. This bidirectional communication allows the sensor data to be passed to the application for processing and storage, while also giving the user control over the network.

The computer application is an easy-to-use multipurpose software with a graphical interface that allows the user to monitor and control the network remotely. It offers all the tools necessary to collect data and record experiments including real-time sensor data visualization and plotting, data processing and storage, multi-camera sensor synchronized video, among others.

Building such a solution required multidisciplinary work, from hardware to software to mechanical design. Since everything had to operate according to our guiding principles, each component was designed from the grounds up. The custom hardware design including electronic design, Printed Circuit Board (PCB) layout, chip selection, and programming interface were all designed to minimize footprint. While the firmware was modified and extended to operate as a bidirectional low-latency, high-throughput network with additional features like node synchronization. The computer application design focused on ease of use and provided practical tools for experimenters through the use of robust frameworks and software libraries. And finally, a careful mechanical design was necessary to retain a small wearable form factor while adding a modular connection system and physical robustness.

2.3 Accomplishments

1. Conceptual design of a distributed sensor platform for wearable instrumentation and sensor exploration.

2. Designed and built hardware including electronics, PCB layout, and board assembly for both sensor nodes and receiver hub with features such as:

   - Very small 13 mm by 18 mm board footprint. (node)
   - On board IMU sensor. (node)
   - Low profile connector-less programming interface. (node)
   - Green and red indicator LEDs for status information. (node)
   - Analog node variant with two 12 bit ADC channels supporting up to two external analog sensors.
   - Output node with two digital outputs for controlling external digital devices.
   - Concurrent charging circuitry for five nodes. (hub)
   - Programmer board for flashing firmware.

3. Designed and coded sections of the firmware running on the nodes and hub to provide basic runtime operations and sensor network features such as:

   - Bidirectional Bluetooth communication.
   - 250 Hz sensor sampling.
   - On-board buffering.
   - Synchronization.
   - Sub 100 ms latency, high throughput data transfer.

- Bidirectional USB computer/hub communication interface.

4. Designed and built a computer application to communicate with the sensor network and provide features such as:

   - Simple and easy-to-use graphical user interface.
   - Data processing and storage in CSV or TXT format with proper headers.
   - Real-time data visualization and plotting thanks to close integration with the KST open source software.
   - Multi-camera synchronized video recording thanks to close integration with the OBS open source software.
   - Command sequence tool to create custom sequences controlling the Output node remotely with event scripting.
   - Trigger feature to execute specific commands automatically.
   - Easy-to-use automated software installer and updater.
   - Cloud integration with data file upload, in-browser visualization, and remote download.

5. Designed, modeled, and manufactured all mechanical aspects of the system including:

   - 3D printed node enclosure with high resilience, minimalist form factor, and LED light guides.
   - Compact 3D printed hub enclosure with USB port, external dipole antenna, and five docking stations for recharging and storing the nodes.

- Innovative connection system using Pogo-style connectors along with small neodymium magnets, used for recharging and attaching to anchor points on the body.
- 3D printed anchor points offering a versatile platform to attach the nodes anywhere on the body, including shoe lace anchors, strap anchors, and custom sensor modules.
- Connection modules of different variations to interface with custom user analog sensors or output digital devices.

6. Validated the system's performance by running a number of test to ensure high reliability and detect potential issues in areas like:
   - Synchronization accuracy.
   - System range.
   - Data loss.
   - Physical robustness.
   - Magnetic connection strength.
   - Charge and discharge times.
   - Application bugs and compatibility.
7. Independent assessment of the system usability and performance by third party experimenters which used it in a real-world setting.
8. Analyzed the independently recorded data to look for behaviour patterns and extract meaningful information, showcasing an example of potential uses of the system.

Chapter 3

Hardware

There are two main hardware subsystems: the Node (and its variants) and the Hub. When designing the hardware of such a system, many parameters have to be taken into consideration, from physical factors such as weight and size to radio communication technologies and battery life. These requirements guide everything from the selection of electronic components to the layout of the PCB.

3.1 Node

The term "wearable" is often used to describe electronic devices that can be worn on the body comfortably without impeding the actions of the wearer. In order to be considered wearable, the nodes have to be lightweight with a very small form factor. However beneficial, reducing the overall size of the node has large implications on the electronic design. Since the largest component in wearable and portable devices such as smartphones and smartwatches is often times the battery, reducing the size of the node restricts the design to smaller less powerful batteries, which in turn affects operation time, power usage, and the choice of wireless technologies. For these reasons, Bluetooth was found to be the

most suitable wireless communication method between the nodes and the hub due to its low power consumption and small physical footprint. Each of the nodes is comprised of a few core subcomponents: a microcontroller, a radio, and power management circuitry. In addition to these, the IMU Node has an Inertial Measurement Unit (IMU) sensor for motion sensing, and the Output Node has an output driver board to control external devices.

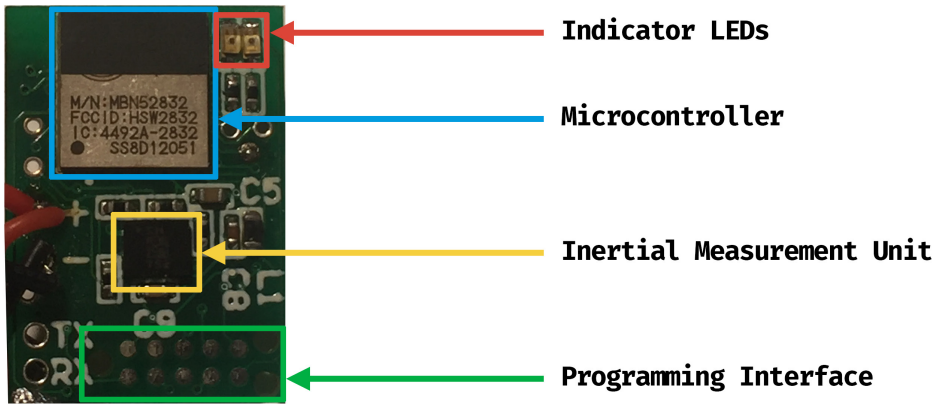


Figure 3.1: The populated Node board with core components highlighted.

3.1.1 Microcontroller

There are a wide selection of Bluetooth microcontrollers on the market that are all suitable for this application. The MBN52832 by Murata [30] based on the NRF52 [31] microcontroller series by Nordic Semiconductor was chosen due to having support for the latest version of Bluetooth 5, its small size at 7.4 x 7.0 x 0.9mm, integrated radio transceiver and on-chip Bluetooth antenna, as well as an extensive Bluetooth API and large amount of documentation and supporting material. This microcontroller also requires minimal external circuitry to operate, reducing its overall footprint even further. The NRF52 microcontroller is built around an ARM Cortex-M4 CPU running at 64MHz, with 512KB of Flash memory and 64KB of RAM [32], sufficient for recording, storing

and transmitting sensor data at high sample rates. Additionally, the microcontroller supports all communication interfaces required with UART for serial communication, I2C for the IMU sensor, and a 12bit Analog to Digital Converter for the Analog Node.

3.1.2 Power Management

In order to power the node and still maintain the size requirement, a very small 40mAh Lithium Polymer (LiPo) battery was chosen, occupying a small volume of 12 x 16 x 5mm and weighting about 2 grams. To reduce and regulate the 3.7 volts from the battery to the 3.3V required by the microcontroller an AP7312 [33] Linear Drop Out (LDO) regulator was used. The enable pin of the regulator was also connected to a MAX16054 [34] switch controller and a flat momentary button that acts as an ON/OFF switch for the whole node, allowing the system to be reset or turned off when not in use. Finally, a small voltage divider circuit measures the battery voltage level through one of the microcontroller's analog pins.

3.1.3 LED Indicators

Each node possesses two small surface mounted green and red LED indicators hooked up directly to the microcontroller's digital output pins and programmed to relay status information to the user.

3.1.4 Node Variants

There are three node variants, all containing the same core components: the NRF52 microcontroller, power circuitry and battery. However, each has additional circuitry to accommodate its function.

IMU Node

To measure linear acceleration and angular rotation, the IMU Node contains the LSM6DS3[35] Inertial Measurement Unit from STMicroelectronics[36]. This system-in-package chip contains a 3D accelerometer and a 3D gyroscope for capturing motion in 6 degrees of freedom. Its low power consumption, fast sampling rate (up to 1.6 kHz), very small footprint (2.5 x 3.0 x 0.83mm), and I2C support made it an ideal sensor for use in the IMU Node.

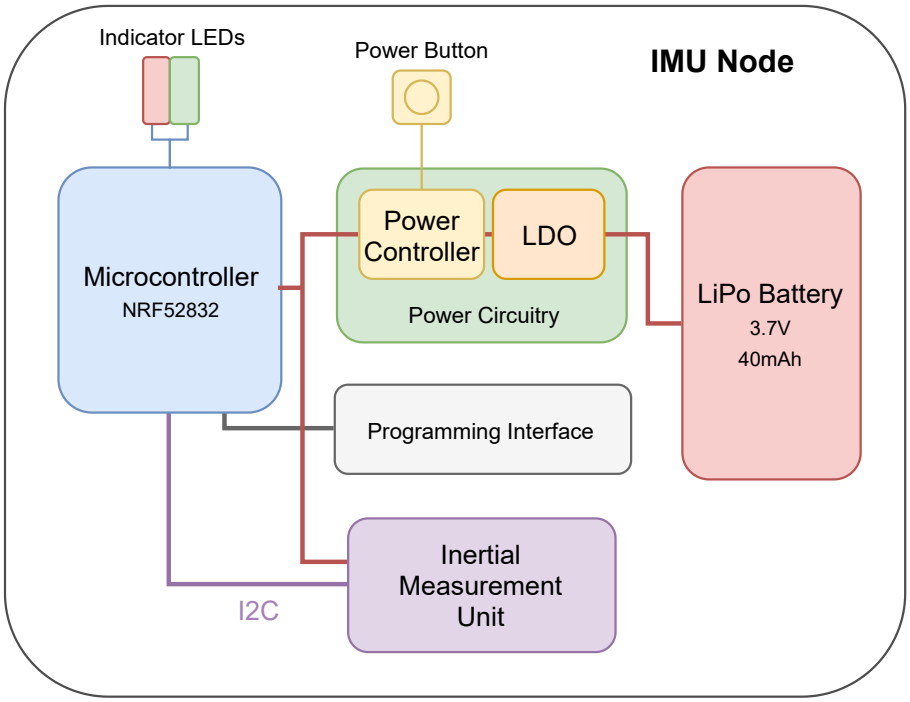


Figure 3.2: Diagram of the IMU node variant with added I2C circuitry for the Inertial Measurement Unit.

Analog Node

The analog node has no additional components on board. Instead, two of the microcontroller's analog input pins, along with two sets of power and ground pins, are connected directly to an external connector. These 6 pins can then be accessed externally by the user through the connection modules which allows the connection of external active and passive analog sensors to the system. The microcontroller's 12 bit Analog to Digital Converter (ADC) digitizes the analog input and incorporates it into the wireless sensor network.

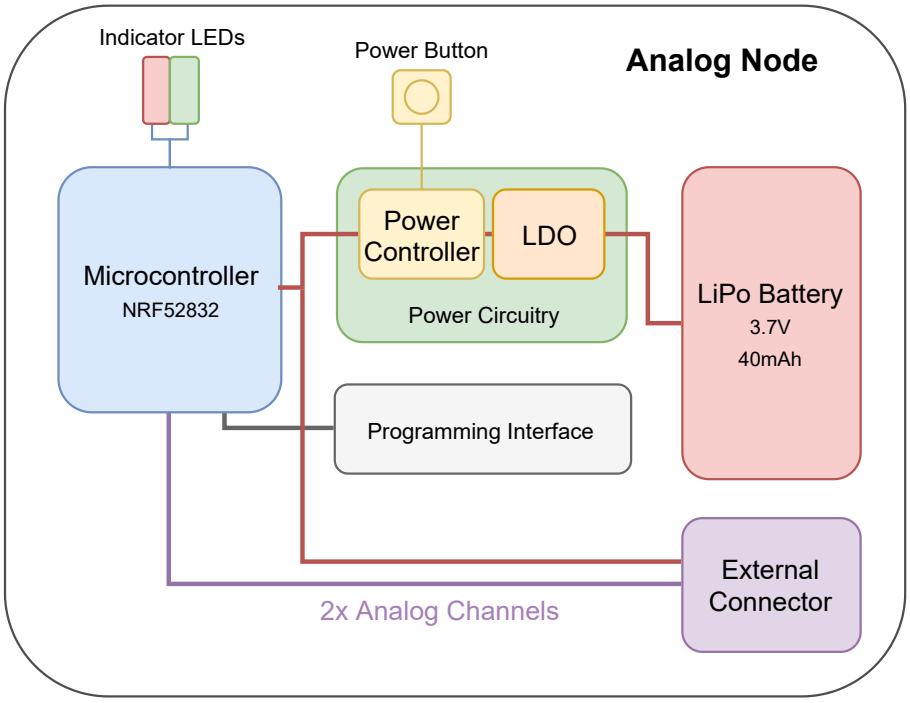


Figure 3.3: Diagram of the Analog node variant which contains an external connector with two analog channels, power, and ground pins.

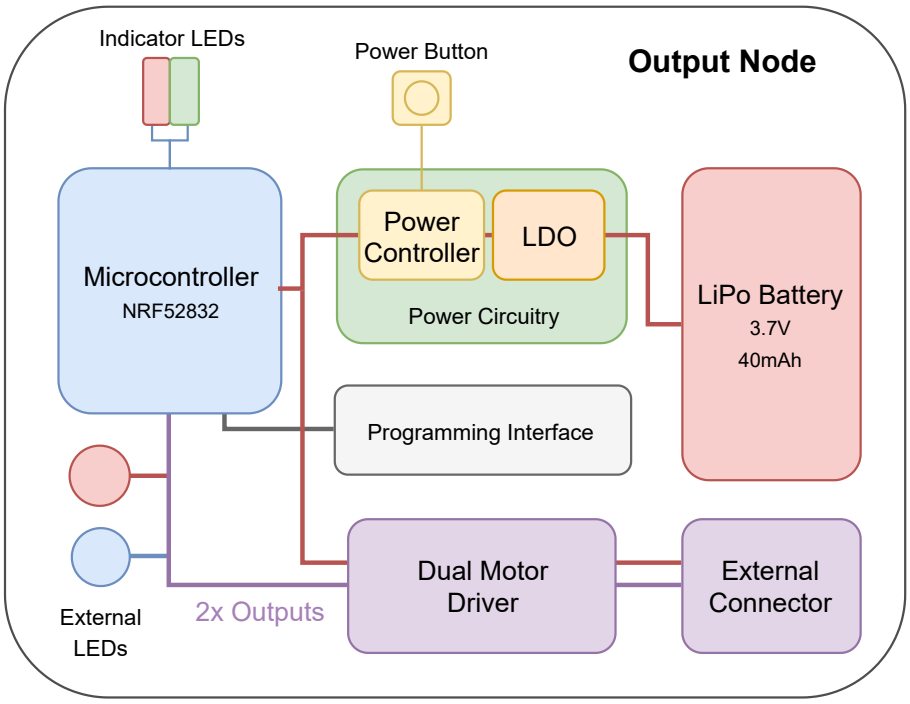


Figure 3.4: Diagram of the Output node variant which contains external red and blue LEDs, a dual motor driver board, as well as an external connector with two digital outputs, power, and ground pins.

Output Node

Unlike other variants, the output node does not record any sensor information from the environment, instead it outputs information into the world by controlling up to two external electronic devices. Such devices include LEDs, buzzers, relays, motors, actuators, haptic feedback devices, TTL systems, and any other devices that can operate with a 3.7V power input. The simplest way of controlling such devices would be to directly connect them between the node's ground and one of its programmable digital output pins. However, this was soon revealed to be unsuitable due to a 30mA maximum

current draw limit from each pin. To remedy that, a dual motor driver board based on the DRV8835 by Texas Instruments [37] was used. This allowed the addition of external power sources, permitting devices to draw current directly from battery or from up to 11V of a user connected external power source. Similar to the Analog Node, an external connector on the back allows the user to connect custom devices using one of the various connection modules. The two connected external devices are color coded red and blue with the Output Node incorporating two additional indicator LEDs used to keep track of the status of each device.

3.1.5 Electronic Design

Once all the components for the node have been chosen, the system schematics were drafted. The reference designs provided by Nordic Semiconductor for the microcontroller and the application designs for each of the subcomponents were used as a baseline for the design. Some schematic component models were retrieved from online repositories such as SnapEDA[38] while others had to be custom designed following the component datasheet. After the addition of passive circuitry with bypass capacitors and pull-up resistors for the data lines, the subcomponent interconnections were implemented and verified before moving on to the PCB layout design.

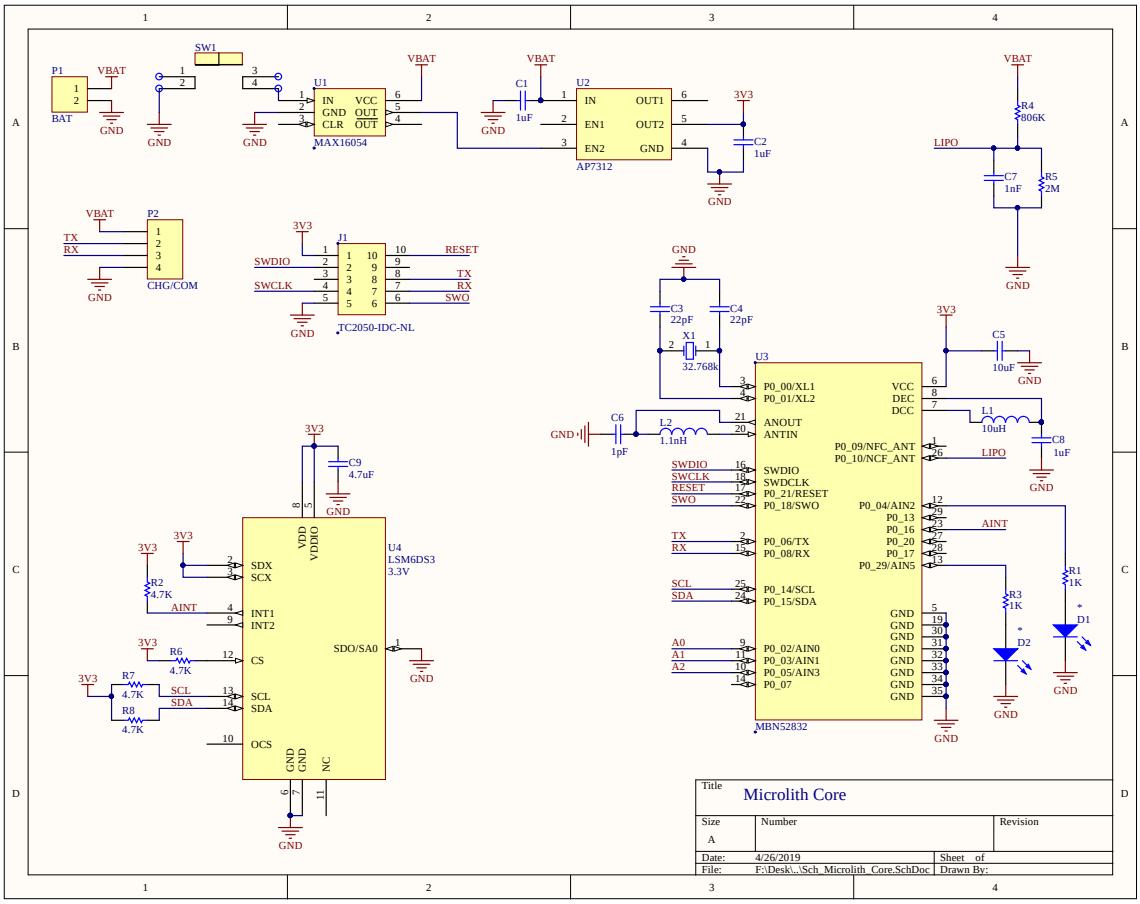


Figure 3.5: Schematics of the node's electronic design. U1: Power controller, U2: LDO, U3: Microcontroller, U4: IMU sensor, J1: Programming interface.

3.1.6 Layout and Board Design

The layout for the node circuit board was similarly implemented with a few key requirements guiding the design. Due to the node's size restriction, the board footprint had to be as small as possible, using the smallest possible components and smallest connection traces allowed by the PCB manufacturer. However, since the boards were to be assembled by hand, the components had to still be large enough to allow for hand soldering. To accommodate that, most passive components like resistors and capacitors were chosen to be of a standard surface mount size of 0402mm. As for the larger active ICs, the smallest available package was chosen while avoiding Ball Grid Array (BGA) packages

which are more difficult to solder by hand. Apart from size restrictions and ease of assembly, it was important that the first stage prototypes be inexpensive to manufacture. Thus the design was limited to a two layer board and by other restrictions on the lower cost prototype service offered by the manufacturer, including trace widths and "via" diameters among other things. Finally, even with these restrictions, the board had to have enough exposed pins for battery, serial, analog, I2C, debugging and programming to allow for certain functionalities and testing.

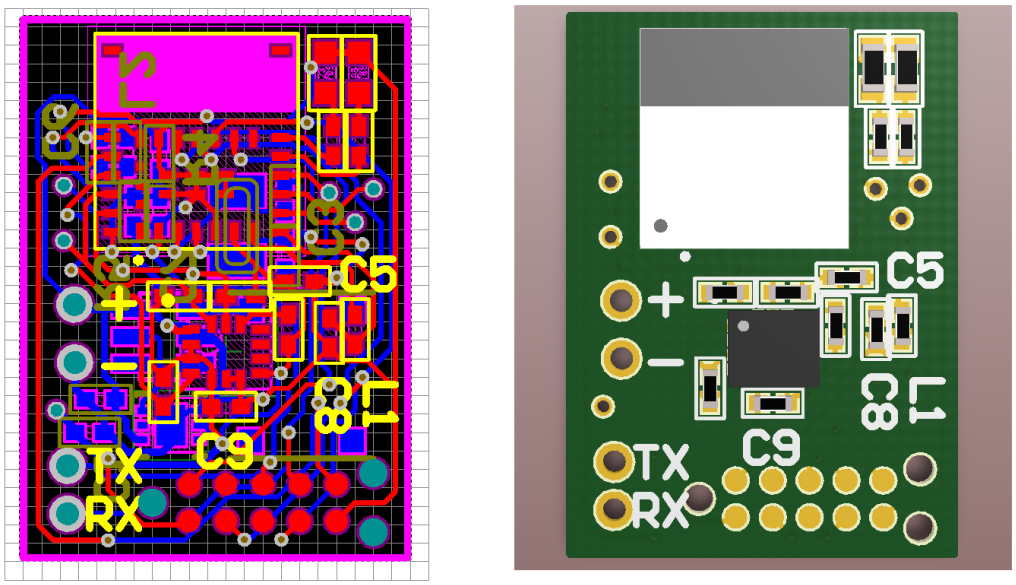


Figure 3.6: PCB layout design of the node.Left: 2D traces corresponding to the different PCB layers. Right: 3D representation of the board.

3.1.7 Programming Interface

One such important set of pins is the programming interface used to flash firmware applications onto the microcontroller. Two of the most widespread programming interfaces supported by the microcontroller are the Joint Test Action Group (JTAG) and Serial Wire Debug (SWD) standards [39]. SWD was chosen due to its minimal requirement of

only four pins: power, ground, Serial Wire Clock (SWCLK), and a bidirectional Serial Wire Data (SWDIO). With these four pins, the microcontroller can then be connected to an SWD programmer such as the J-Link by SEGGER [39] and programmed by a computer through USB.

In addition to the SWD pins, a UART serial interface with three pins: TX, RX and RESET were exposed, to allow for a simple method of communication between the computer and the microcontroller after flashing. This was used as a secondary programming interface with Arduino as well as a way to relay runtime debugging information.

Physically, these pins only need to be connected to external devices during programming, debugging and testing. A permanent connector, such as a micro USB, which would not be used after the development stages would uselessly occupy a large space on the board, greatly increasing the size of the nodes. For this reason, we explored multiple temporary interface solutions including temporarily soldered header connectors and board sections that could be cut off after initial development.

The most suitable solution was found to be the Tag-Connect [40] system that allows for ten flat pogo-style pins contact pads to be used as a temporary wired connection. Its reusability, minimal footprint, flat surface, and cost made it the ideal candidate for this project.

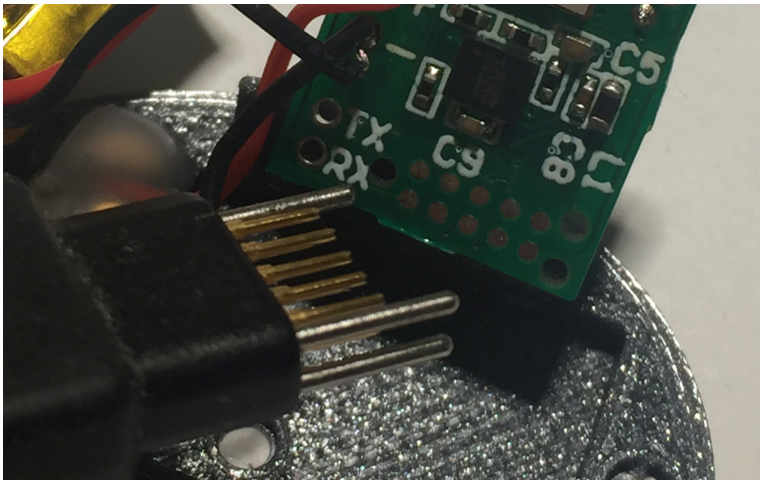


Figure 3.7: The Tag-Connect[40] connection system used for the programming interface. It uses pogo-style pins for communication and power, and three guiding rods to align properly.

3.2 Hub

As the name implies, the hub is the central unit to which all the nodes connect to. It occupies three main roles in the system, each with corresponding circuitry and components. First, it acts as a wireless receiver, collecting sensor data sent from the nodes. Second, it is the interface between the nodes and the computer software. Third, it is a charging and storage station for the nodes.

The hub is a stationary wired unit without the stringent size and weight restrictions of the nodes, resulting in fairly different design decisions, such as a large receiver antenna and a wired USB connection for data transfer and power. Unlike the electronic design of the nodes which was built from the ground up, the hub incorporates an off the shelf microcontroller board alongside a custom built one. Without size restrictions, incorporating a prebuilt component greatly accelerated development and provided features out of the box that would otherwise need to be reimplemented.

The Bluefruit Feather board by Adafruit [41] was chosen for this role as it incorporates many of the same core components found in the nodes, including a microcontroller

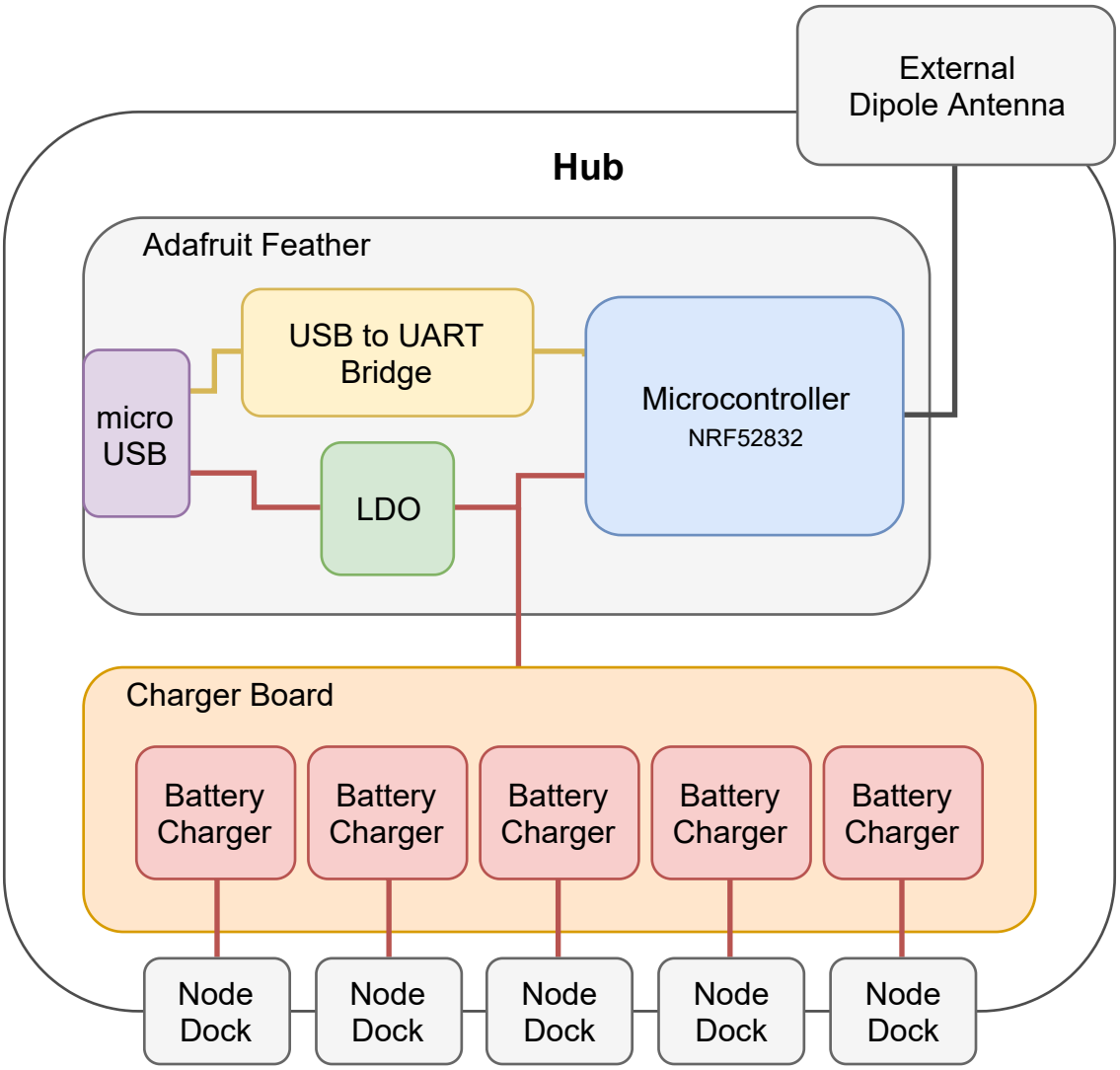


Figure 3.8: Diagram of the Hub including the off the shelf Adafruit Feather, as well as the custom charger board with five node chargers.

based on the same NRF52832 architecture, the MDBT42 by Raytac [42]. This ensured that the same codebase could be used across devices without modifications while still benefiting from the MDBT42's upgradable antenna. In addition, the Feather board comes prebuilt with a micro USB connector as well as a Serial-to-USB chip, simplifying the communication interface between the hub and the computer.

3.2.1 Wireless Receiver

The hub is the central unit in the Bluetooth sensor network. By using a microcontroller based on the same NRF52832 chipset found in the nodes we ensure that both subsystems have the same Bluetooth 5 implementation and reduce any chances of protocol incompatibilities. Since size is not a constraint for the hub, a large dipole antenna was added to dramatically increase the range of the sensor network. This was done by replacing the external chip antenna found on the Bluetooth module by a u.FL connector, connected to a 2dBi dipole antenna through a u.FL to RP-SMA pigtail.

3.2.2 Computer Interface

As a mediator between the sensor network and the user, the hub has to communicate with the computer software to interchange data, user commands, status information, and general debug messages. This is done through the CP2104 USB/UART bridge by Silicon Labs [43] found on the Bluefruit Feather board. This IC bridges the UART pins of the NRF52832 microcontroller on one side and the computers USB port on the other, establishing a USB 2.0 interface between the two. And since the Bluefruit Feather board includes a micro USB connector, a standard micro USB to USB cable is used to connect the hub to the computer.

3.2.3 Charging Station

Aside from its role in the sensor network, the hub also doubles as a charging station for the nodes. To achieve that, in addition to the Bluefruit Feather board, the hub contains a custom made charger board populated with five BQ2404 Lithium Polymer battery chargers from Texas Instruments[44]. Each charger is rated at 1 Amps which is capable of charging the node's batteries in less than an hour. The chargers are fed directly from

the USB power input into the hub.

3.2.4 Electronic and Board Design

The schematics and layout for the hub were designed similarly to those of the node. However, since this board operates in conjunction with the Bluefruit Feather, it is much simpler in design as it only contains the five chargers and corresponding circuitry. Two additional components were also incorporated into the design which was only populated on a single board exemplar that acted as a development unit. A UART/USB bridge (CP2104 similar to the one found on the Bluefruit Feather) with a micro USB connector was added along with connector headers for the programming interface which was used to program and test the nodes during development. The programming interface is used to connect the Tag-Connect cable between the node's board and the J-Link programmer, the micro USB and UART/USB chip then bridges the node's board to the computer's USB port for Arduino development and runtime debugging. The straightforward design and lack of size restrictions allowed for a much simpler one layer board layout, simplifying manufacturing and assembly.

3.2.5 Manufacturing and Assembly

To reduce manufacturing complexity and cost, the board designs for a complete system were combined onto a single board, with one hub layout surrounded by five nodes with so called "mouse-bites" breakaway bridges in between each design. Ten boards were then manufactured and shipped by PCBCart [45] over the course of 10 days. After delivery, the various components and ICs for three complete systems were purchased from DigiKey [46] and assembled and soldered by hand.

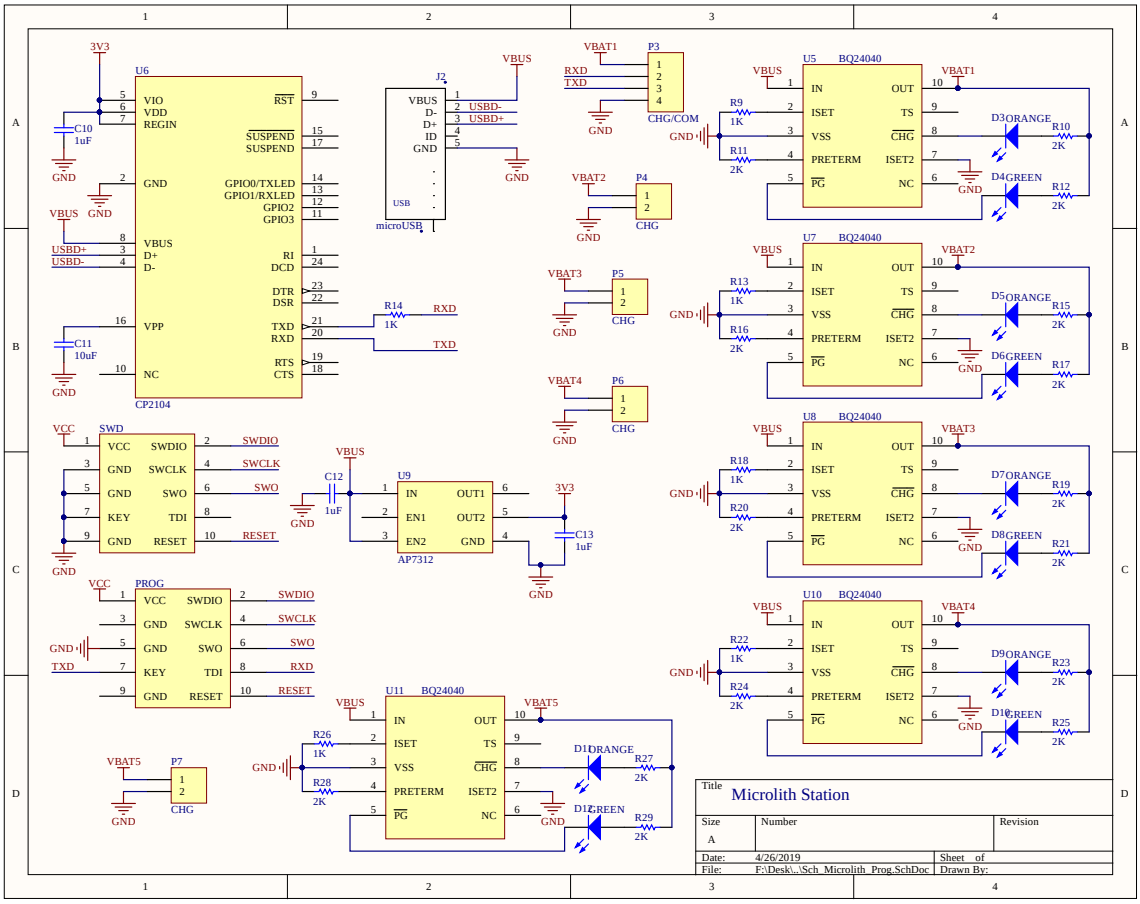


Figure 3.9: Schematics of the hub's electronic design. U5, U7, U8, U10, and U11: Chargers, U9: LDO, P3-7: Charging pins. There are also components used only on the programmer board including U6: Serial-USB bridge, PROG and SWD: Programming headers.

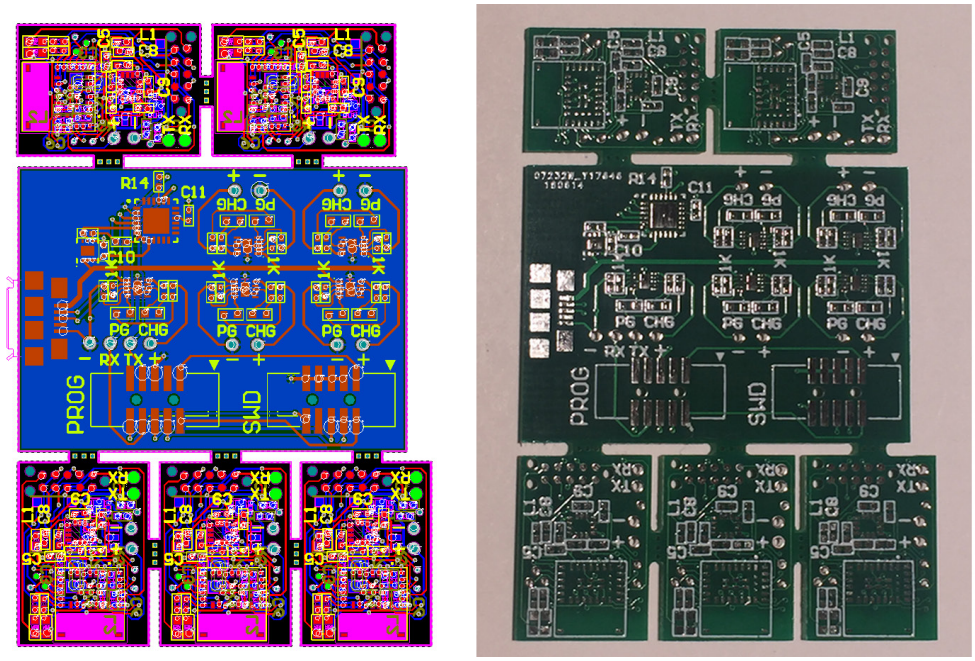


Figure 3.10: Final PCB layout incorporating both the node and hub for manufacturing, with "mouse-bites" for easy removal. Left: Layout design. Right: Resulting PCB manufactured by PCBCart.

Chapter 4

Firmware

As a microcontroller based on the ARM architecture, the NRF52832 supports a multitude of compilers and software frameworks including one for the Arduino ecosystem [47]. As a framework focused on rapid prototyping, Arduino greatly simplifies and accelerates the initial firmware development by providing a simple and easy to use Integrated Development Environment (IDE) along with a wealth of third party libraries, documentation, and support.

4.1 Firmware Stack

The Arduino support is enabled by a software stack provided by Adafruit and consisting of multiple software layers. The lowest layer in the firmware stack is the SoftDevice S132 by Nordic Semiconductor [48]. This layer consists of the bootloader required to initiate program execution, the System-on-Chip libraries which deals with the low level functionalities of the microcontroller, as well as the Bluetooth 5 Protocol Stack that includes libraries for controlling the Bluetooth radio. On top of this layer, and communicating through API calls, resides FreeRTOS [49], a free real-time operating system

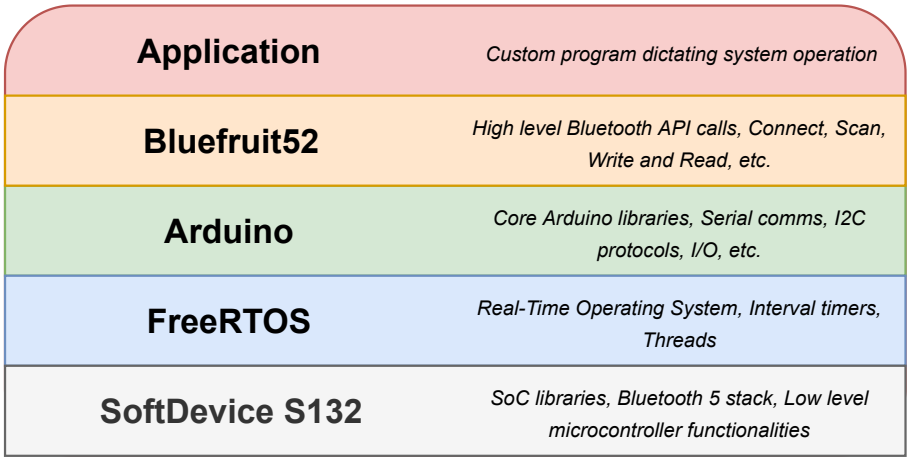


Figure 4.1: Firmware stack layers. Each layer builds on the one below it and goes from the lowest level code at the bottom of the stack, towards the user application at the top.

for microcontrollers. FreeRTOS adds the ability to run threads and interval timers and is essential for Arduino operation. Support for the Arduino IDE and functions is then provided by the core Arduino library stack, which includes libraries for serial communication, I2C protocols, delays, I/O controls, and others. In addition to the Arduino core libraries, the firmware stack includes a third party library called Bluefruit52 provided by Adafruit [50], that abstracts many of the Bluetooth functionalities and provides a much more simple and user friendly API to execute Bluetooth commands such as connect, scan, write, and read. Finally, at the top of the stack runs the user defined Arduino application, dictating the high level operation of the program. It is where most of the custom software is written and is the only part that varies between the different devices in the system.

4.2 Firmware Modification

During development, some sections of the firmware stack had to be modified to better fit the scope of the project. Such modifications were heaviest near the top of the stack as most missing features could be implemented with API calls, and without modifying

much of the underlying codebase. The bootloader, containing the SoftDevice S132 stack, was provided as a .hex binary file and had no option to be modified. Regardless, there was little reason to modify it as much of its functionality was required as is. Slight modifications were required to the FreeRTOS layer of the stack, mostly to the configuration files. These changes increased the tick rate of the system from 1kHz to 32kHz providing higher resolution timers and resulting in more accurate Bluetooth transmission intervals.

Since many desired functionalities were missing from the Bluefruit52 library, additional code was added to support features like the ability to change the Bluetooth Maximum Transmit Unit payload size, changing the Bluetooth Physical Layer settings, reducing connection interval delays and other Bluetooth features and configuration parameters. In addition to the Bluefruit52 Bluetooth library, new third party sensor libraries had to be introduced to allow for specific sensor functionalities, including libraries for the LSM6DS3 IMU sensor.

Finally, since the application layer is project dependent, it was created from scratch for each for the nodes and hub to operate as desired. This layer includes functionalities to interface with Bluetooth devices such as connection and disconnection, scan and transmission data events, sensor interfaces to read data from sensor devices, as well as peripheral operation functions to read the battery level, control the LEDs, etc.

4.3 Flashing Firmware

After writing the firmware source code, several steps were required to compile the code, write it to the microcontroller's flash memory and successfully boot and run the Arduino program. Since the microcontroller's flash is initially empty and contains no code, it is impossible for the computer and Arduino IDE to detect or communicate with it. As such, the first step in programming the microcontroller is to flash the bootloader. This

is done through the Serial Wire Debug hardware interface with the aid of a J-Link programmer. The programmer connects to the programming software running on the computer through USB on one side and to the microcontroller through the SWD pins on the other. The programmer then proceeds to copy the binary bootloader file to the flash memory of the microcontroller. After a power reset, the bootloader code executes and the microcontroller is recognized by the computer, ready to be programmed by the Arduino IDE through a serial COM interface. Before compiling and uploading the Arduino code, it is essential to modify the Arduino configuration files to include the so called "Board Specific Definitions" and match the pins from the board's hardware design to the ones assigned by Arduino at startup, including the various input/output pin and data lines such as I2C. The previous steps were only required the first time a new microcontroller was programmed. Any subsequent programming can be done by simply pressing the upload button in the Arduino IDE, this makes prototyping and testing easy and fast, resulting in greatly accelerated development which is the primary advantage of the Arduino ecosystem.

4.4 Node Firmware

The role of the nodes is to collect data samples continuously from the connected sensors, compress and packetize it into a Bluetooth payload, and transmit it to the central hub at preset intervals. The IMU and Analog nodes share similar firmware since their functions are alike. The Output node, on the other hand, differs significantly both in its firmware and the role it plays in the sensor network. The following section discusses some of the key functions found in the node firmware.

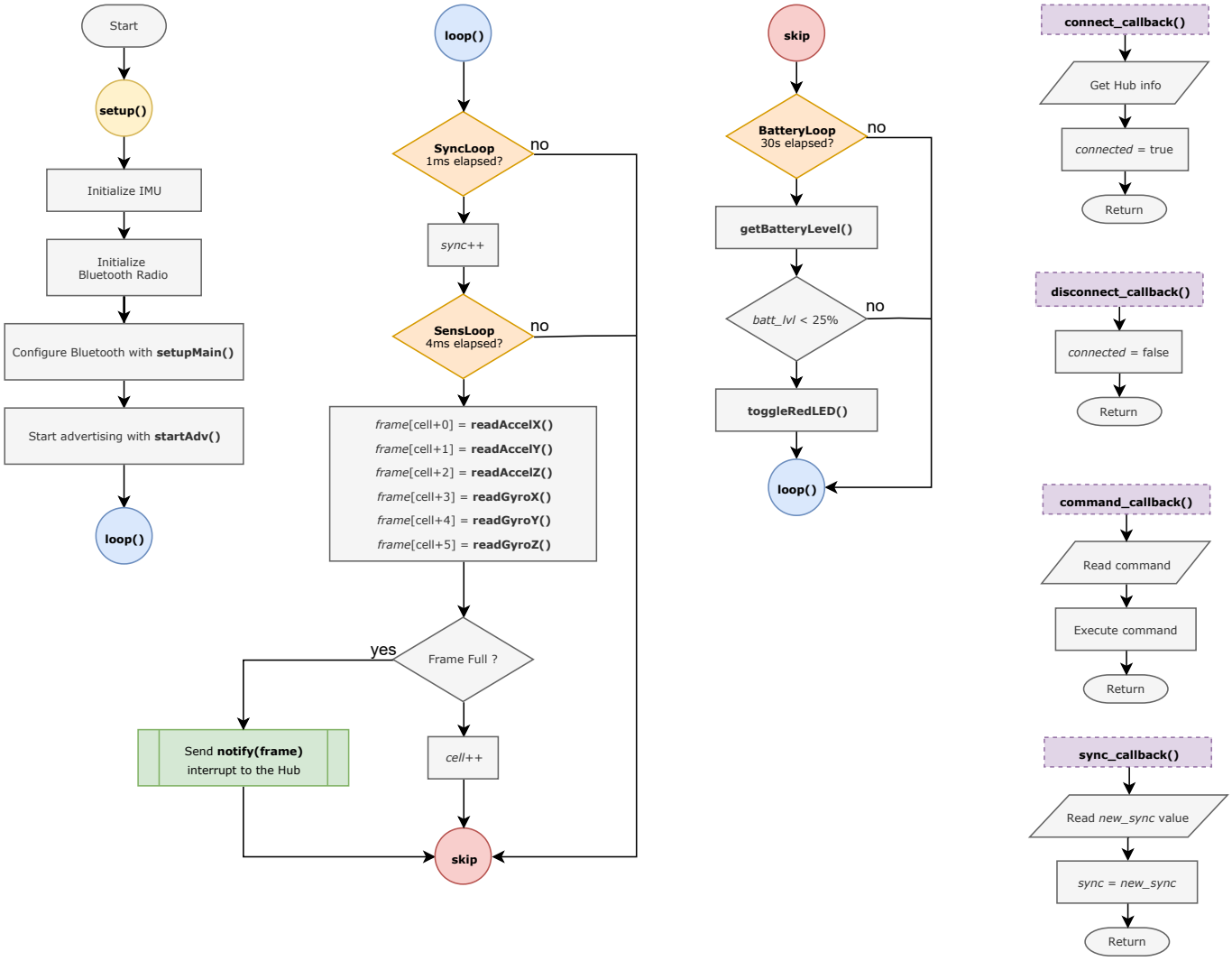


Figure 4.2: Flowchart of the node's program. The yellow and blue circle represent the two Arduino functions **setup()** and **loop()**. The orange diamonds represent the three main non-blocking loops, including the **SyncLoop**, **SensLoop**, and **BatteryLoop**. The green rectangle is an interrupt sent to the hub whenever data is ready to be sent, while the purple rectangles are local interrupts that are triggered by hub events.

4.4.1 Arduino Dependent

void setup()

The setup function is called at the start of any Arduino program. In the node, the setup function is used to check for the sensor and initialize it, setup the Bluetooth configuration such as name and transmit power settings, setup the Bluetooth data structures and protocols, and start advertising the node on the network.

void loop()

The loop, as its name implies, acts as an infinite while loop where most of the real-time functionalities reside. This main loop is divided into multiple small, non-blocking loops based on repeating timers that take care of various functions including synchronization and time tracking, filling the sensor data buffer, and updating the battery level information.

4.4.2 Bluetooth Specific

There are two type of Bluetooth functions: direct call and callback functions. Direct call functions instructs the Bluetooth radio to execute certain commands directly, while callbacks are software interrupts that execute in response to a predefined event.

Direct Calls

void setupMain()

Called once at the start of the program `setupMain()` sets up all the configurations, services and characteristics data structures required for the Bluetooth protocol.

void startAdv()

This function configures and starts the advertising of the node on the Bluetooth network, causing the hub to recognize and connect to it after performing its scan.

Callbacks

void connect_callback(...)

The connect callback is called as soon as the hub sends a request to connect to the node, at which point the node simply records the hubs information and connects to it.

void disconnect_callback(...)

Similarly to the connect callback, the disconnect callback occurs when the hub sends a request to disconnect from the node. This triggers the node to reset and wait for a new connect request.

void command_callback(...)

The command callback is a method to enable two way communications between the node and the hub. If a command payload is written by the hub to the node's Bluetooth "command" characteristic, the node reads the payload and executes a function based on that command, such as toggling one of the two LEDs.

void sync_callback(...)

The sync callback updates the local `sync` variable to match the one sent by the hub. This is the primary way of keeping the nodes synchronized to the hub, and thus to one another.

4.4.3 General Functions

`int getBatteryLevel()`

In order to get the latest battery level, the analog pin connected to a voltage divider hooked up to the battery is read. The ADC value recorded is then converted to voltage and mapped to a percentage scale.

4.4.4 Loops

There are three core non-blocking loops that iterate as long as the node is connected to the hub. Each loop is defined by an iterator, a fixed interval, and a counter.

```
if(micros()-tick>=interval){
    tick = micros();
    // tasks
}
```

`micros()` is an internal Arduino function that calculates the number of microseconds elapsed since the start of the program by using the RTOS tick count and clock speed. This number is then compared to the one recorded during the previous iteration, if the difference exceeds the desired interval, the following tasks are executed and the iterator is set to the current time. Otherwise, the tasks are skipped until the next check without blocking the program.

Sync Clock Loop

This loop keeps track of an internal synchronization clock `sync` that ticks every millisecond. It is different from the Arduino's internal clock, in that it can be modified

by an external signal through the `sync_callback` function. This allows the hub to resynchronize all connected nodes to the same clock value remotely.

Sensor Reading Loop

The sampling rate of the sensors is defined by a 4 millisecond interval loop (250Hz). Every iteration, a reading is taken from up to 6 sensor channel, and added to a frame buffer. Instead of transmitting it at a fixed rate, the data is sent asynchronously once the buffer is full, this ensures that each packet sent by the radio is always of the same size. Once the buffer is full, a copy of the data is sent to the hub through a Bluetooth "notify" interrupt, the buffer is emptied, and the loop starts again. By keeping the sensor loop running on the node's local clock, we ensure that the inter-sample duration are consistent, resulting in high signal consistency even across multiple payloads, and minimal disruption in case of lost packets.

Battery Loop

Every 30 seconds the battery level is checked, if it is below 25% the red LED is turned on to indicate to the user that the node is almost out of power.

4.4.5 Variables

Bluetooth Datastructures

Bluetooth information is stored inside a nested structure of Services and Characteristics, each with a unique ID. These Characteristics can be read from or written to by the hub, enabling communication over the Bluetooth network. In the nodes, one main Service is present, containing three core Characteristics, one for the synchronization signal, one for the sensor data payload, and one for the command request. Reading or writing to

one of those Characteristics triggers its associated callback.

Timer Iterators

A variable of type `long` is used to keep track of the microseconds passed between each iteration of the various loops. These variables are updated at the start of each loop by setting it to the microseconds elapsed since the start of the program. There are three timer iterators or Ticks, one for each loop: `sensTick`, `syncTick`, and `batteryTick`.

Data Payload

In order to transmit Bluetooth data at the most efficient rate, the data payload size has to be lower than the Maximum Transmit Unit of 247 bytes. To achieve this, the various sensor channels (the 6 accelerometers and gyroscope axes for the IMU Node, or the two analog channels for the Analog Node) of multiple successive samples have to be serialized. The payload is represented by a byte array of 240 bytes. Each sensor channel occupies two bytes with a maximum of 6 channels resulting in 12 bytes per sample. After each sensor reading loop, each sample is appended to the previous one, until the array is full after 20 samples, at which point the entire frame is transmitted. Each payload sent thus contains 20 samples of up to six 16-bit sensor channels.

4.5 Node Variants

The IMU and Analog variants are both input nodes and differ only slightly in the sensor reading section of the code. In order to read the accelerometer and gyroscope data, the sensor has to be initiated during the setup stage. After that, during the sensor reading loop each of the six sensor channels are read sequentially to form one sample which is then stored in the frame buffer. The Analog node is similar, but instead of reading

data from an IMU sensor, it reads from two sensor devices connected to the analog pins. The data from the analog devices is stored in two of the 6 available channels in the data payload. Since only two of the six available channels are occupied, the Analog node transmission scheme is a prime candidate for throughput optimization in future works. The Output node differs significantly since its role in the sensor network is fairly different. Instead of sending sensor data, the Output node waits on a signal from the hub to control LEDs and connected external devices. It does not have any sensor reading or data transmission methods and instead uses an expanded version of the `command_callback` method to receive information from the hub.

4.6 Hub Firmware

In comparison to the nodes, the hub's firmware is somewhat more complex as it has to both interface with the computer software on one end and act as the Bluetooth Central device on the other.

4.6.1 Bluetooth Functions

void startScanning(...)

This function starts scanning for surrounding Bluetooth devices, as soon as a node is discovered the `scan_callback` is called to start connection procedures.

void disconnect_all()

In order to reset the system, it is required to disconnect from all connected nodes.

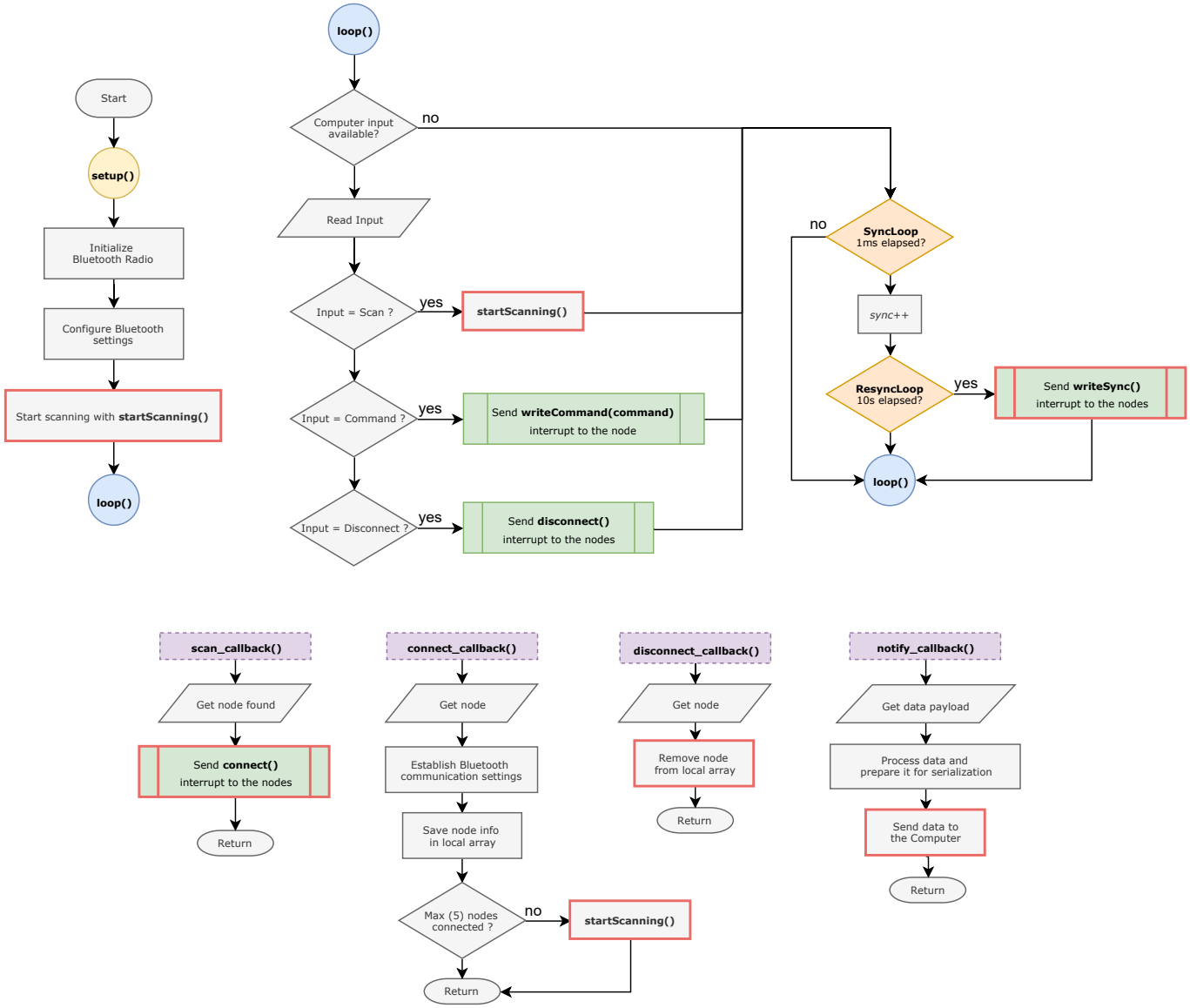


Figure 4.3: Flowchart of the hub's program. The yellow and blue circle represent the two Arduino functions **setup()** and **loop()**. The orange diamonds represent the two main non-blocking loops **SyncLoop** and **ResyncLoop**. The green rectangles are interrupts signals sent to the nodes, while the purple rectangles are local interrupts that are triggered by node events. Rectangles with a red outline indicate information or data sent to the computer.

void writeCommand (. . .)

This function is used to send specific commands to a node with a desired id and is linked to the node's own `command_callback` interrupt. This is similar to a Remote Procedure Call (RPC) as it allows the node to execute specific functions based on the hubs request.

void writeSync ( )

In order to resynchronize the nodes, the `sync_data` variable is written to each of the nodes, triggering the nodes `sync_callback` interrupt.

4.6.2 Callbacks

void scan_callback ( )

As soon as a new node is discovered the `scan_callback` is executed. This is followed by an attempt to connect to the newly discovered node, followed by a `connect_callback` on a successful connection.

void connect_callback ( )

After a connection is established, the `connect_callback` takes care of setting up the connection parameters. First, the hub checks if the connecting node is valid and ready to communicate, after which the Bluetooth communication settings are established, such as negotiating the Physical Layer (PHY) mode and the Maximum Transmission Unit (MTU) payload size. Following that, the node is given an identification number, its variant type is recorded, and the information is stored in the local devices array. Finally, if the maximum number of connected devices (5) is reached, the scan is stopped, otherwise a new 3 seconds scan is performed using the `startScanning` function.

void disconnect_callback()

If one of the nodes disconnects (turned off, out of range, etc.), the `disconnect_callback` is triggered. The node's ID is recorded and removed from the list of connected devices.

void notify_callback()

This callback occurs whenever a node sends a data payload to the hub. The hub receives the payload as a 240 bytes array, appends to it the ID and type of the node that sent it, and pipes it to the serial interface.

4.6.3 Serial Communication

The serial interface between the computer software and the hub allows for two way communications between the devices. Computer inputs are commands sent from the computer software to control the hub, the nodes, or the sensor network as a whole. While hub outputs are responsible for sending sensor data and network status information back to the computer.

Computer Inputs

The computer software interfaces with the hub via the UART protocol for bidirectional communications. The hub listens for any available data on its serial interface after each main loop, if a message containing a command ID is identified, the hub executes the corresponding command. This allow the user to directly control the sensor network in various ways such as starting a new scan, toggling the red LED of individual nodes, reading the battery levels, controlling the Output Node, or disconnecting from all nodes.

Hub Outputs

The primary data sent by the hub to the computer software is a payload containing the sensor data. The payload sent to the software contains the main 240 bytes of sensor data arriving from the node, metadata information on the sensor type and ID, followed finally by a delimiter indicating the end of the payload. Aside from the data payload, the hub also sends status updates following various events such as synchronization, scanning, and connection events. This notifies the software and helps the user keep an eye on the status of the network, and be alerted whenever errors occur.

Chapter 5

Computer Application

At its most basic level, the role of the computer application is to collect and record data from the sensor network. However, thanks to the versatility of computer software, many features were added to greatly improve the user experience, open up the system to new applications, and overall provide much value to the user. There are a few key features that the application allows in addition to simply collecting and recording the sensor data. These include synchronization between the individual nodes, keeping track of connected sensors, real-time visualization and plotting of the incoming sensor data, simultaneous video capture from external cameras, and controlling the Output Node remotely.

The computer application was built using the Qt framework [51], an open source collection of libraries and tools that facilitates the development of applications with graphical user interfaces and provides interfaces with low level hardware. The framework was used to provide a native looking user interface design, control external computer processes, and better communicate through the USB serial interface.

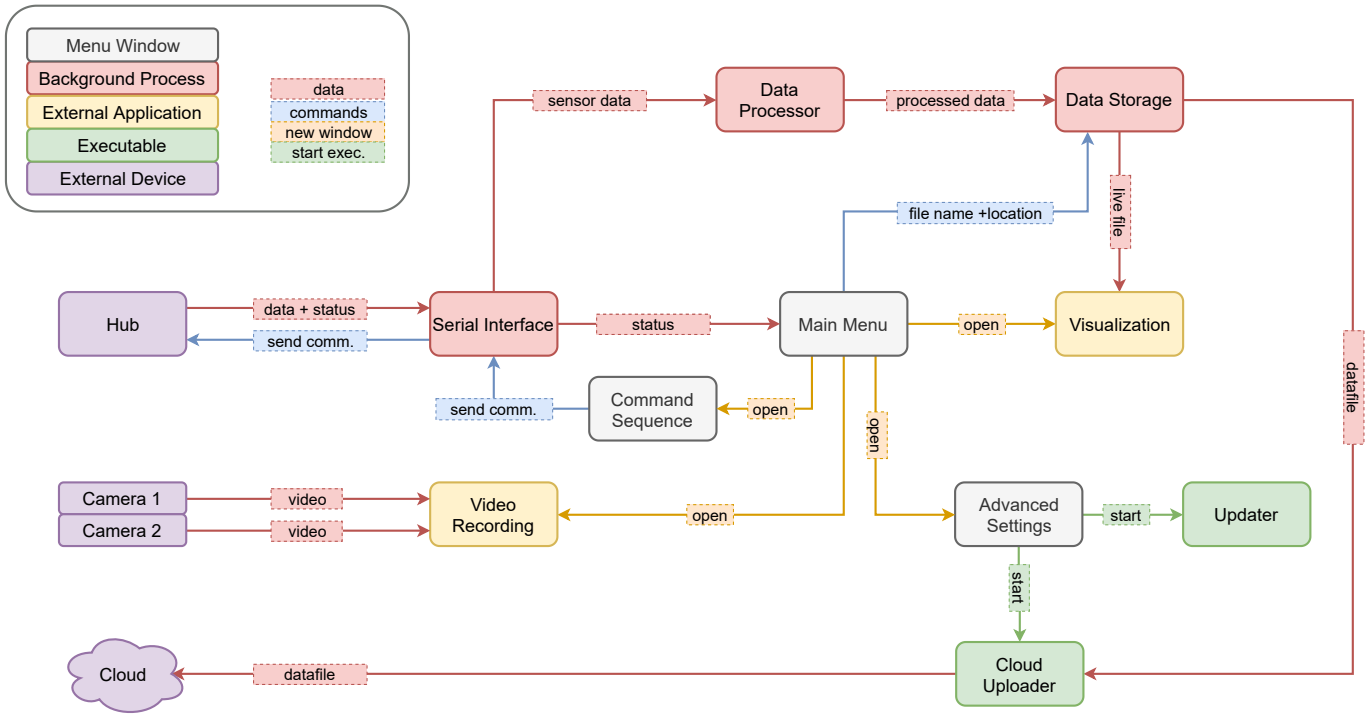


Figure 5.1: Application overview diagram showing all menu windows, background processes, external applications, executable commands, as well as devices external to the computer. The arrows represent the different paths taken by the collected data, as well as the user commands and button presses.

5.1 Main Menu

When the application is first started, the user is greeted by a simple user friendly main menu that offers all the basic functionalities to control the system.

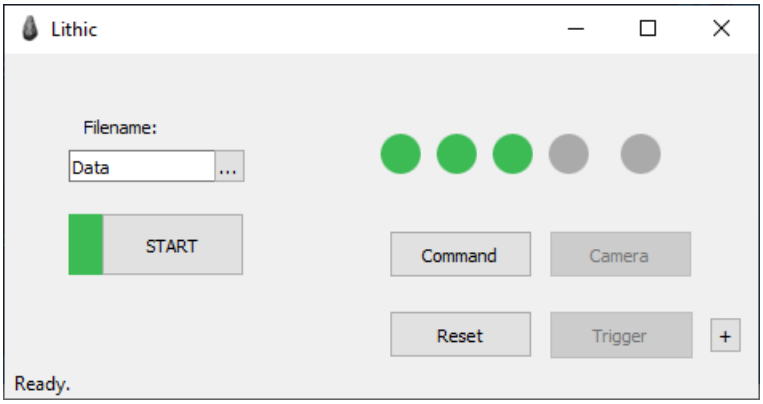


Figure 5.2: The Main Menu window that greets the user on application startup.

If all the nodes are powered on and the hub is connected to the computer, the user can quickly start a new experiment by simply typing the experiment name in the designated text box and clicking on the start button. This starts recording sensor data from the network and opens the real-time visualization window. In addition to starting a new experiment, the user can check on the status of the various connected nodes by looking at the five colored disks representing each of the nodes. The color of the node indicates its status: gray for nodes that have not been connected to the network, green for the ones that are connected, yellow for nodes with a low battery level, and red for nodes that have lost connection to the network. Right clicking on any of the nodes provides the user with their battery levels, while left clicking on a node will highlight it in the software and physically by turning on the node's red LED. This is useful to identify which node represents which disk in the application. The main menu also contains buttons for restarting the sensor network, opening the command menu to control the Output node, starting any connected cameras, enabling the trigger feature, as well as a

small button to open the advanced settings menu. The main menu also has a status bar at the bottom of the window which updates the user on the status of the network (ie: Recording...), notifies them of various events (ie: Started scan.), and expresses errors that could help with troubleshooting (ie: Hub not found.)

5.2 Serial Interface

It is important for the application to properly communicate with the hub to receive sensor data and send commands. To do so, all serial devices connected to the computer are first scanned at the start of the application. The COM port associated with the hub is then identified by its name and description, and a communication link between the hub and the application is established. At this point, any message that needs to be sent to the hub can be sent through this link, and whenever a data payload or status message is sent by the hub it is received and processed by the application.

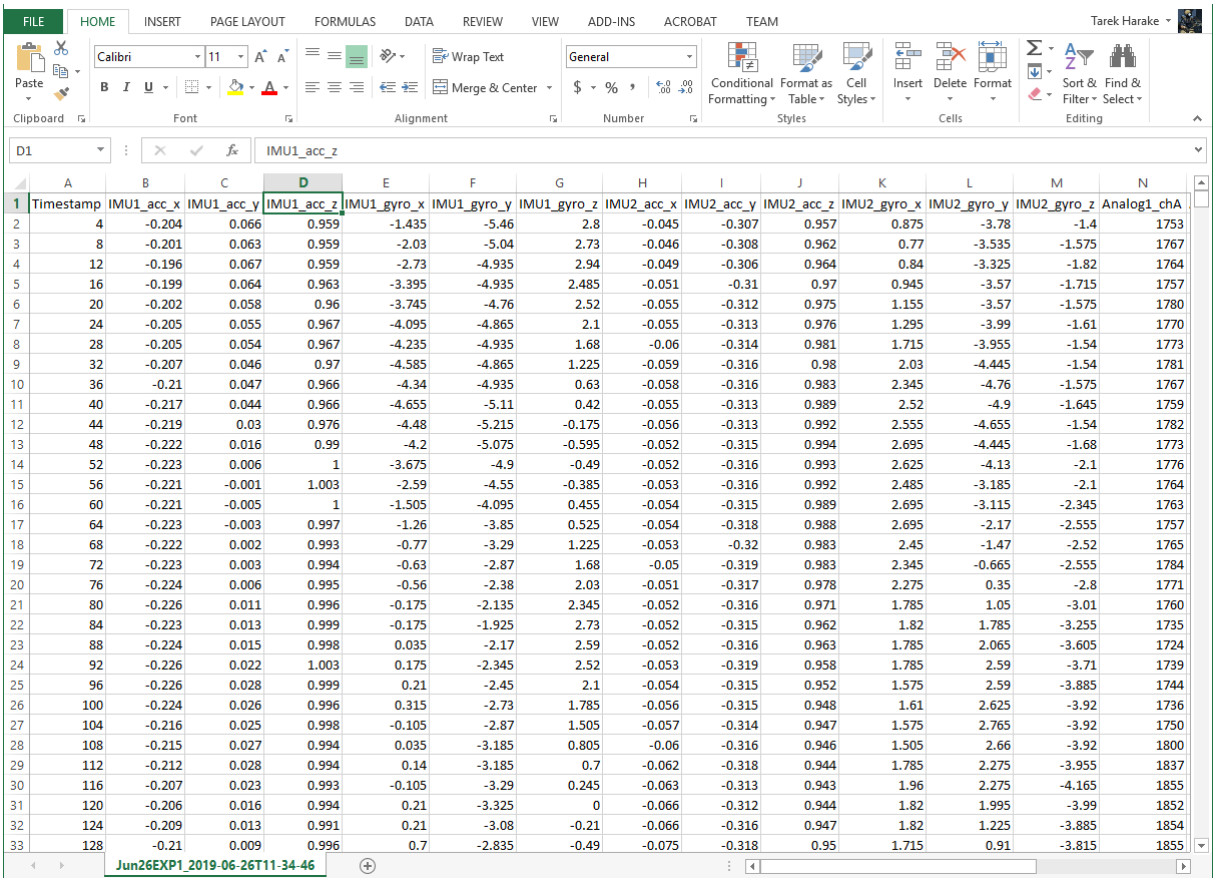
5.3 Data Processor

Once a data payload is received by the application, it is the sent to the data processor where the data is stored in memory, synchronized, decompressed back to sensor information and saved to file, or prepared for visualization.

5.4 Data Storage

When a new experiment is started a subdirectory is created with the experiment's name, within which exists the data file in TXT or CSV format with the name of the experiment as well as the date of recording. Any video footage recorded is also stored as an MP4 in the same folder, sharing the data file's name. In addition to saving the data recorded

during an experiment, data is also continuously stored in a “live.txt” file that keeps appending new information as long as the program is running. This is used to feed the plotting system and create real-time visualizations, as well as a backup file in case an interesting event occurred outside of a recorded experiment. Each recorded data file contains a header with a list of sensor and node names, followed by comma separated values where each column represents a single sensor channel and each row a sample.



Timestamp	IMU1_acc_x	IMU1_acc_y	IMU1_acc_z	IMU1_gyro_x	IMU1_gyro_y	IMU1_gyro_z	IMU2_acc_x	IMU2_acc_y	IMU2_acc_z	IMU2_gyro_x	IMU2_gyro_y	IMU2_gyro_z	Analog1_chA
4	-0.204	0.066	0.959	-1.435	-5.46	2.8	-0.045	-0.307	0.957	0.875	-3.78	-1.4	1753
8	-0.201	0.063	0.959	-2.03	-5.04	2.73	-0.046	-0.308	0.962	0.77	-3.535	-1.575	1767
12	-0.196	0.067	0.959	-2.73	-4.935	2.94	-0.049	-0.306	0.964	0.84	-3.325	-1.82	1764
16	-0.199	0.064	0.963	-3.395	-4.935	2.485	-0.051	-0.31	0.97	0.945	-3.57	-1.715	1757
20	-0.202	0.058	0.96	-3.745	-4.76	2.52	-0.055	-0.312	0.975	1.155	-3.57	-1.575	1780
24	-0.205	0.055	0.967	-4.095	-4.865	2.1	-0.055	-0.313	0.976	1.295	-3.99	-1.61	1770
28	-0.205	0.054	0.967	-4.235	-4.935	1.68	-0.06	-0.314	0.981	1.715	-3.955	-1.54	1773
32	-0.207	0.046	0.97	-4.585	-4.865	1.225	-0.059	-0.316	0.98	2.03	-4.445	-1.54	1781
36	-0.21	0.047	0.966	-4.34	-4.935	0.63	-0.058	-0.316	0.983	2.345	-4.76	-1.575	1767
40	-0.217	0.044	0.966	-4.655	-5.11	0.42	-0.055	-0.313	0.989	2.52	-4.9	-1.645	1759
44	-0.219	0.03	0.976	-4.48	-5.215	-0.175	-0.056	-0.313	0.992	2.555	-4.655	-1.54	1782
48	-0.222	0.016	0.99	-4.2	-5.075	-0.595	-0.052	-0.315	0.994	2.695	-4.445	-1.68	1773
52	-0.223	0.006	1	-3.675	-4.9	-0.49	-0.052	-0.316	0.993	2.625	-4.13	-2.1	1776
56	-0.221	-0.001	1.003	-2.59	-4.55	-0.385	-0.053	-0.316	0.992	2.485	-3.185	-2.1	1764
60	-0.221	-0.005	1	-1.505	-4.095	0.455	-0.054	-0.315	0.989	2.695	-3.115	-2.345	1763
64	-0.223	-0.003	0.997	-1.26	-3.85	0.525	-0.054	-0.318	0.988	2.695	-2.17	-2.555	1757
68	-0.222	0.002	0.993	-0.77	-3.29	1.225	-0.053	-0.32	0.983	2.45	-1.47	-2.52	1765
72	-0.223	0.003	0.994	-0.63	-2.87	1.68	-0.05	-0.319	0.983	2.345	-0.665	-2.555	1784
76	-0.224	0.006	0.995	-0.56	-2.38	2.03	-0.051	-0.317	0.978	2.275	0.35	-2.8	1771
80	-0.226	0.011	0.996	-0.175	-2.135	2.345	-0.052	-0.316	0.971	1.785	1.05	-3.01	1760
84	-0.223	0.013	0.999	-0.175	-1.925	2.73	-0.052	-0.315	0.962	1.82	1.785	-3.255	1735
88	-0.224	0.015	0.998	0.035	-2.17	2.59	-0.052	-0.316	0.963	1.785	2.065	-3.605	1724
92	-0.226	0.022	1.003	0.175	-2.345	2.52	-0.053	-0.319	0.958	1.785	2.59	-3.71	1739
96	-0.226	0.028	0.999	0.21	-2.45	2.1	-0.054	-0.315	0.952	1.575	2.59	-3.885	1744
100	-0.224	0.026	0.996	0.315	-2.73	1.785	-0.056	-0.315	0.948	1.61	2.625	-3.92	1736
104	-0.216	0.025	0.998	-0.105	-2.87	1.505	-0.057	-0.314	0.947	1.575	2.765	-3.92	1750
108	-0.215	0.027	0.994	0.035	-3.185	0.805	-0.06	-0.316	0.946	1.505	2.66	-3.92	1800
112	-0.212	0.028	0.994	0.14	-3.185	0.7	-0.062	-0.318	0.944	1.785	2.275	-3.955	1837
116	-0.207	0.023	0.993	-0.105	-3.29	0.245	-0.063	-0.313	0.943	1.96	2.275	-4.165	1855
120	-0.206	0.016	0.994	0.21	-3.325	0	-0.066	-0.312	0.944	1.82	1.995	-3.99	1852
124	-0.209	0.013	0.991	0.21	-3.08	-0.21	-0.066	-0.316	0.947	1.82	1.225	-3.885	1854
128	-0.21	0.009	0.996	0.7	-2.835	-0.49	-0.075	-0.318	0.95	1.715	0.91	-3.815	1855

Figure 5.3: Example of saved data file opened in Excel. The data is automatically placed in column and rows, with the first row containing the header identifying the each sensor channel and the first column showing the timestamp for each sample with 4ms increments.

5.5 Visualization

One of the advantages of a real-time sensor network is the ability to visualize the sensor data as events are occurring. This allows the experimenter to better understand the effects of certain actions on the sensors, as well as provide real-time feedback to the subject wearing the system. The open source plotting tool KST [52] was used to plot the data on the screen. This is done by feeding the live data from the application directly into KST and assigning the different headers to different subplots in the KST window using a predetermined template file. Aside from its use as the visualization platform, KST offers many abilities that could greatly benefit the experimenter's workflow, including built-in filtering capabilities and a highly customizable layout. Whenever the user starts recording, KST is automatically started with a preassigned template and the data is immediately visualized without any additional input from the user. The default startup template could also be changed to fit an experiment requiring a specific plot layout, filtering, or other custom features.

5.6 Video Recording

For many experiments, the data recorded from sensors alone might not give the whole picture and might be hard to correlate with physical events. A video camera setup recording footage simultaneously could prove very useful and provide much more context than simple sensor data. For this reason, the application incorporates a feature to connect two or more USB cameras and record video that is synchronized with the captured sensor data. This is done by interfacing to a custom modified version of the open source video recording platform Open Broadcaster Software (OBS) [53]. The modifications made allow OBS to be used in the background as a video rendering and recording software, while the application acts as a remote to start, stop, and save the recorded

Figure 5.4: KST visualization program. The node data is plotted in real-time in each of the subplots. The first three rows represent the three IMU nodes with the accelerometer data in the first column and the gyroscope data in the second, each curve representing one sensor channel, three for each accelerometer and three for each gyroscope. The bottom left and right subplots would contain the Analog node channels and Output node commands respectively (these nodes are not connected in this instance). The user has the option to change the template in the Advanced Settings menu to have any type of display or visualization for custom needs.

video. The use of OBS also opens up the possibilities for customized recordings that better suit the experiment. This includes features like adding more camera inputs, overlaying data and plot lines on top of the recorded video, computer screen capture, remote live streaming, among many other.

On startup, the application detects any connected USB cameras and enables the "Camera" button in the main menu, pressing it opens a resizable window showing the camera preview. Starting a new experiment with the preview window open will start recording both the sensor data from the network as well as the video captured by any connected cameras and saves it as an MP4 file.

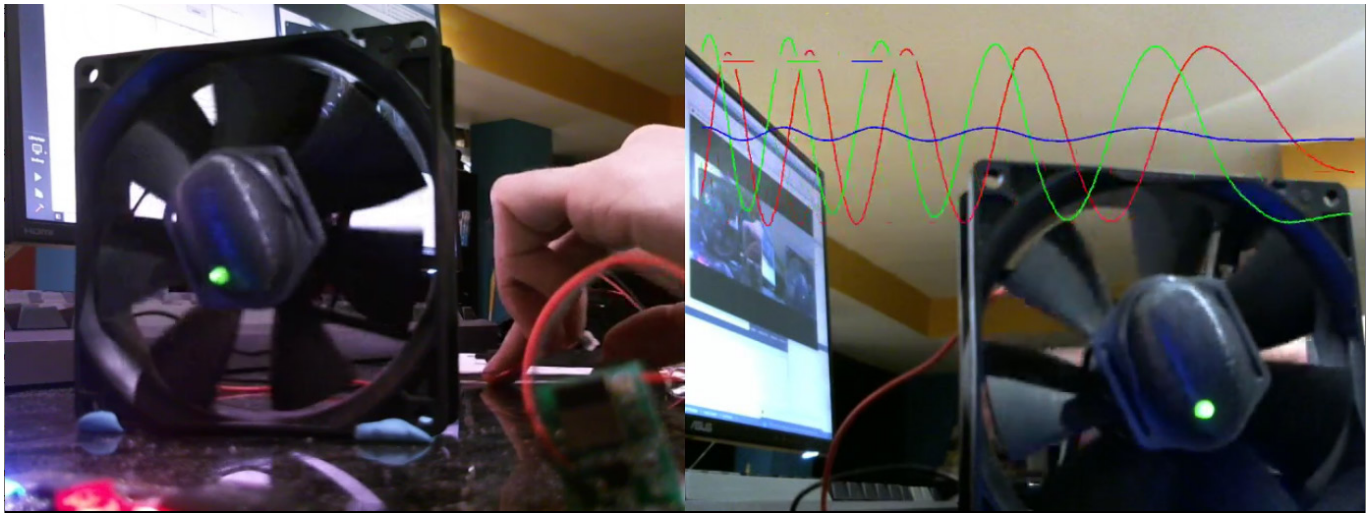


Figure 5.5: Example video recorded with two connected cameras. The video shows an IMU node on a snap anchor attached to a small fan. The accelerometer data from the node is overlaid on top of the video in the top right corner displaying the motion as the node is turning with the fan. This ability to overlay synchronized data plots in real-time over the live video feed greatly increases the versatility of the system.

5.7 Command Sequence

The command sequence menu is a way for the user to programmatically control the Output node by designing sequences for the node to follow. Each of the two output devices supported by the Output node is represented by a red or blue color. The user can then either manually toggle the output devices on or off by pressing the button with the corresponding color, or program a sequence of toggles with predetermined intervals using a command sequence script entered in the text field. The user can then visualize the sequence as a series of blue and red bands corresponding to the different toggle events. Clicking the start button will then start recording while playing the sequence until it is complete. This allows the user to design specific experiments with predetermined lengths and sequences of events.

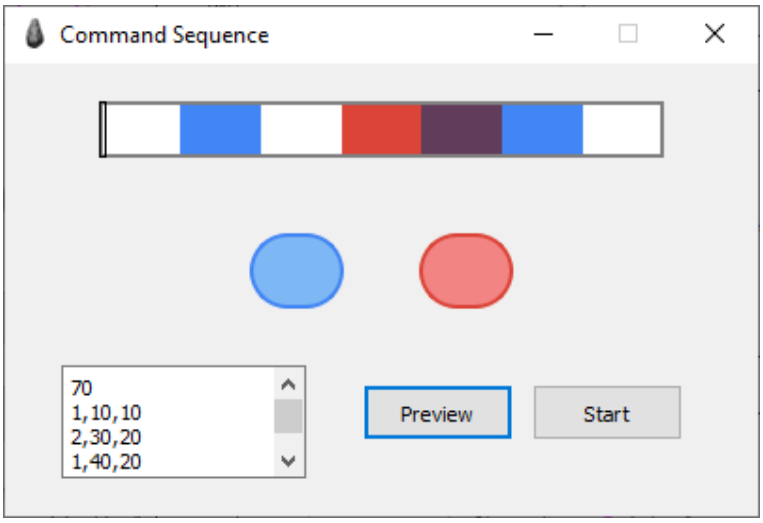


Figure 5.6: Command Sequence window. The two large buttons can toggle the outputs on the connected Output node. The top band represents the coded sequence written in the text box on the bottom left. The script is written with each line representing a timed toggle start and end times, the first number represents the channel to be toggled, the second number is the start of the toggle in increments of x100ms while the third number represents the duration of the toggle in x100ms.

5.8 Advanced Settings

The software also contains some advanced or uncommon features and settings that do not belong in the main menu, but could nevertheless be useful in some situations where the experimenter requires more control. In order to keep those features without overwhelming the user experience, we opted to hide such settings and features behind the advanced menu button. These included settings such as using zeroes instead of empty space for missing data, saving data files under a CSV format instead of TXT, using custom visualization templates, and setting up a remote network data storage. The advanced settings menu also includes some features such as enabling the trigger functionality, which executes specific commands if one of the analog channels met a trigger condition, sending the data to the cloud, opening a custom OBS instance, or checking for application updates.

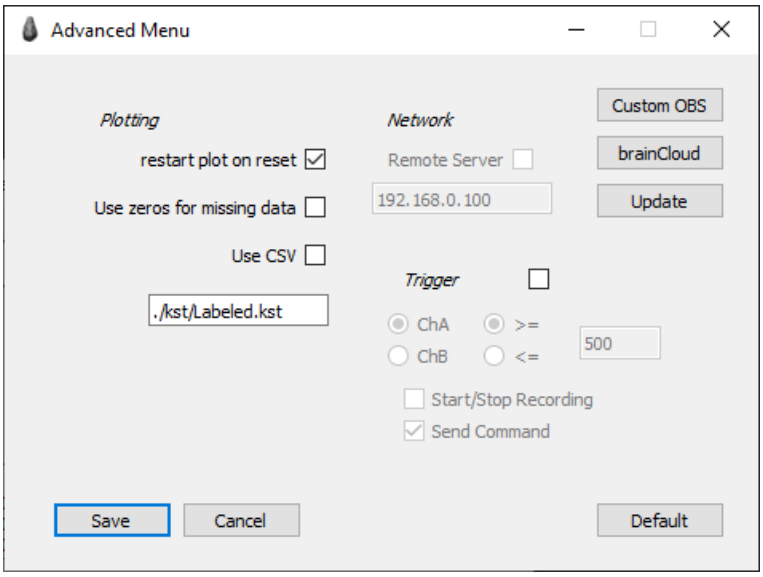


Figure 5.7: The Advance Menu window. The user can modify the application's behaviour by changing these settings and saving them. The saved settings will remain even after application restart until the user presses the Default button.

5.9 Installer and Updater

In order to simplify the installation procedure, the application is packaged inside an installer wizard that takes care of installing the application onto the host computer including asking for a desired install directory and creating a desktop shortcut. In addition to the installer, the application also includes an updater that can fetch the latest updates from a remote server and update the application with minimal user intervention. This ensures that the user always has the latest version of the software, and that the system can be constantly improved even after initial deployment.

5.10 Cloud Integration

In addition to storing the recorded data locally, the software is capable of sending data files to the brainCloud database, a Backend-as-a-Service developed by BitHeads [54].

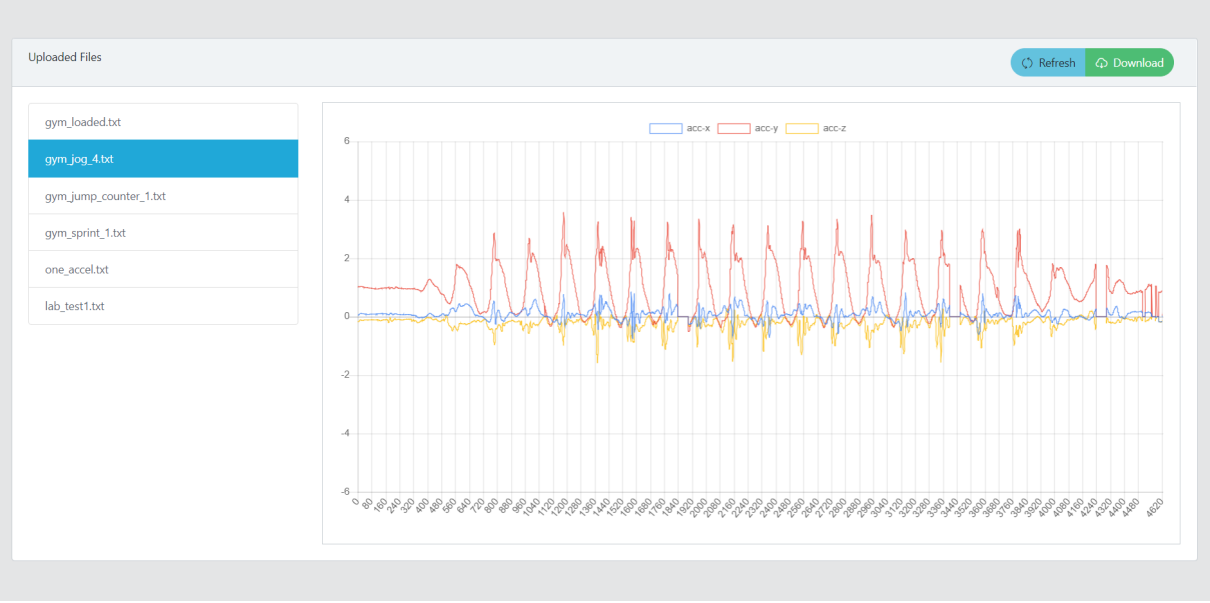


Figure 5.8: WebApp with in-browser visualization. After a file is uploaded, it is shown in the tab list on the left. Clicking the file allows the user to visualize the data within the browser, as well as downloading it using the green button on the top right.

The database, in conjunction with a custom built webserver written in JavaScript and HTML5 allows users to upload their files to the cloud, and access them later on from the browser. This enables the user and their colleagues to navigate through recorded experiments, visualize them in the browser, and download the data files from anywhere and on any device with an internet connection.

Chapter 6

Mechanical Design

6.1 Design Criteria

There were some industrial and mechanical design requirements that were essential to making a wearable system, especially one that is distributed over multiple locations on the body. These requirements guided the design throughout its development. One such core condition was to reduce the size of the nodes as much as possible while still maintaining enough physical robustness to sustain drops and harsh environments.

6.2 Framework

The mechanical design of the nodes went through multiple iterations to refine its external shape, connector wiring, PCB and battery placement, and other aspects to ensure that the node functions well as a modular wearable device, while maintaining a clean looking exterior. Rapid prototyping methods were employed to greatly accelerate the iterative process and allow for much more fine tuning and design freedom. The models were first designed in the Fusion360 CAD software by Autodesk [55], and then 3D printed using a

desktop Fused Deposition Modeling (FDM) printer, the Prusa i3 Mk3 [56]. This allowed multiple modifications and variations to be designed, manufactured, and tested within very short time spans.

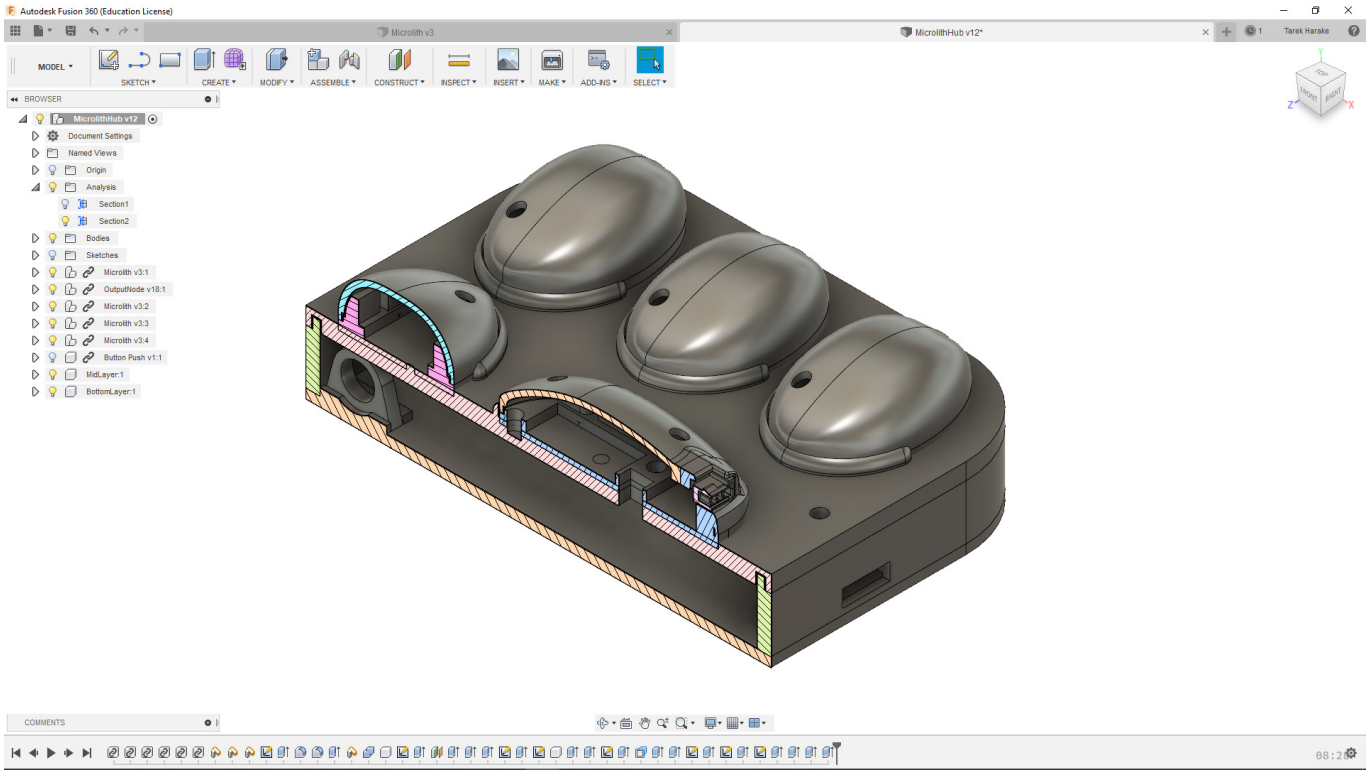


Figure 6.1: Hub and node enclosures designed in Fusion360, a complete CAD software suite with features such as parametric modelling, multi-body assembly and joins, and section analysis tools (shown here).

6.3 Magnetic Connector System

Ease of use and low setup time were primary goals during the design. Having up to five nodes distributed around the body would traditionally greatly increase the complexity of setting up the system, or even modifying the setup during an experiment (to recharge a node or modify its position). To remedy this, a simple yet robust multipurpose magnetic connector system was developed. Using a pogo-style connector for a cable-less connec-

tions and three small neodymium magnets at the bottom of each node, the connector system acts as an interface to the outside world.

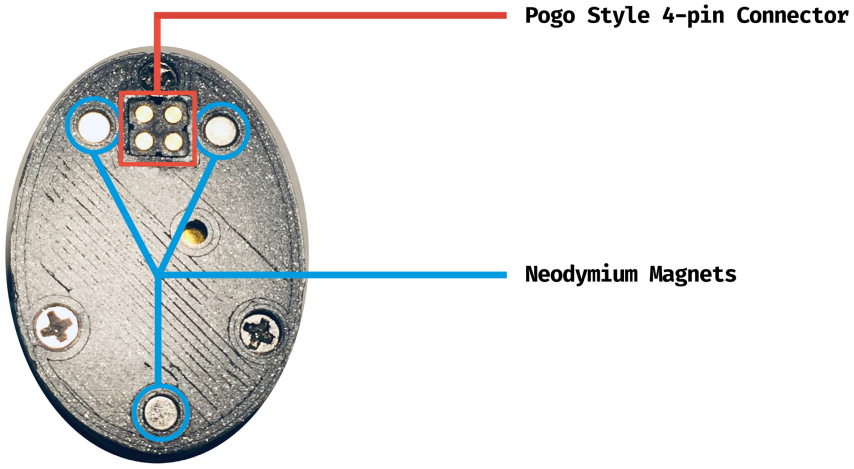


Figure 6.2: The Magnetic Connection System. Comprised of three neodymium magnets with 3mm diameter that ensure a strong connection to the hub and the anchor points, and a square 4 pin pogo-style female connector for charging.

The connector system enables the nodes to dock with the hub and recharge through the pogo connectors without the need to plug in any charging cables, reducing the node's overall size due to the lack of external connectors such as a traditional Micro-USB. Another role of the magnets is to attach to "snap anchors" on the wearer's body. Instead of strapping the nodes directly to the body, snap anchors are worn instead. These are 3D printed docks with the same magnetic coupling method that allows the nodes to securely snap to them, allowing the wearer to quickly attach and detach the nodes from their body by simply sliding them off, without removing the snap anchors. The magnets are strong enough to keep the nodes attached during experiment activities while still allowing the wearer to easily detach them. This greatly decreases the duration of setup changes between experiments, testing on multiple subjects, recharging the nodes, and

simplifies the overall use of the system. Different location specific snap anchor designs were built for various applications, such as ones for the chest and head with a slot for inserting a Velcro or textile straps, and ones with shoelace holes for nodes measuring data from the foot. Yet another application of the magnetic connector system is to attach to external sensor modules in the case of the Analog node. One example is an ECG sensor module with a chest strap to which the Analog module could be attached. Furthermore, external modules could in the future be designed to communicate with the attached node through the pogo pins, allowing for the addition of modular sensors without the need for external wired connections.

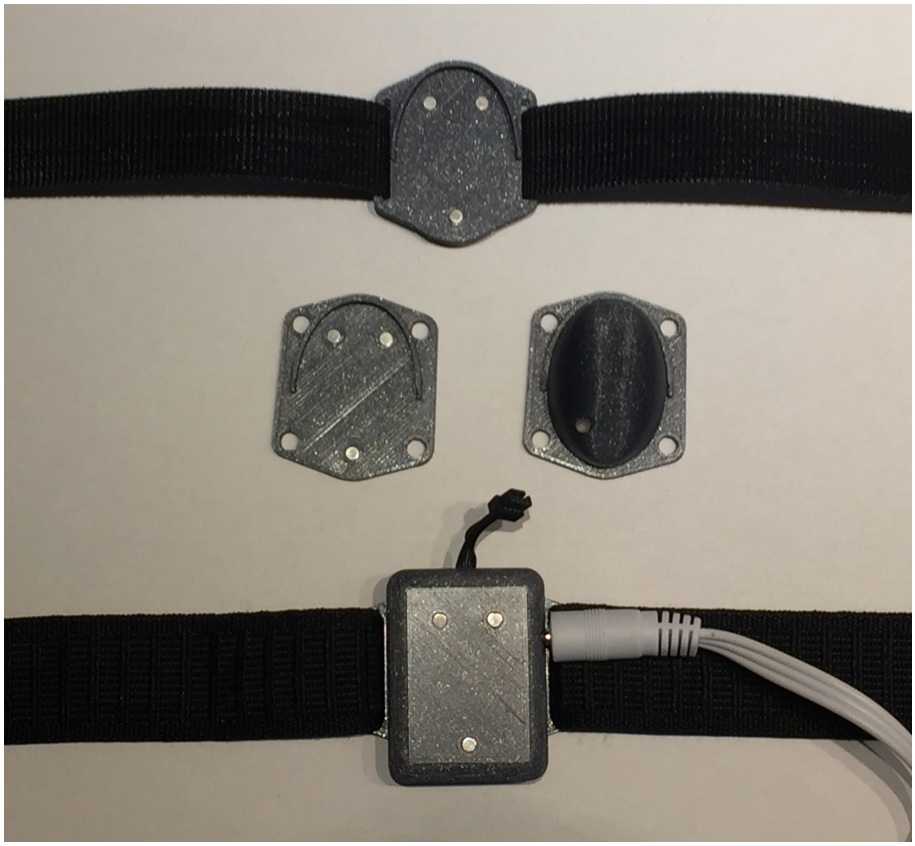


Figure 6.3: A variety of snap anchors. Top: The strap anchor features a velcro or fabric band allowing the node to be placed on various parts of the body including head, hands, arms and legs. Middle: The shoe anchors incorporate shoelace holes for a secure fit on the wearer's foot. Bottom: The ECG module is a special anchor with a built-in ECG sensor that communicates to the Analog node and a flexible belt for easy placement around the chest.

6.4 Nodes

After multiple iterations, the final IMU and Analog node design snugly fits the PCB and the battery and weights around 7 grams and occupies a volume of around 35 x 25 x 15mm. The Output node is slightly larger due to the additional motor driver board and weights around 10 grams, and with a volume of around 45 x 25 x 15mm.

The Analog node supports up to two different analog sensor inputs. It does so by exposing pins for the power, ground, and two analog input pins connected to the microcontroller's ADC. To expose all these pins to the outside, a 4 pin connector is built into the back of the node. This allows a small cable to snap securely to the node and connect to the external sensors. Many variations were tested and the ultimate design allows the connector to be easily accessible and well secured to the node without changing the node's form factor and keeping it compatible with the magnetic connector system.

The Output node required a similar design to allow the addition of an external 6 pin connector for connecting external output devices. However, even more changes to the Output node were required, in addition to the external connector, two additional LEDs with light guides were added near the back. These two red and blue LEDs are indicators that turn on whenever the external device connected on their side is active. The LEDs also allow the Output node to be used as a standalone light indicator device in certain experiments, without the need for external devices or discrete LEDs.

Due to the presence of a second PCB board, the Output node is the only one with a different form factor and footprint, requiring a more elongated device. However, the front section was specifically designed to match the rest of the nodes, and uses the same magnetic connector system, making it compatible with the various snap anchors and modules.

Figure 6.4: Attached to the back of the Analog node, the small 4 pin rectangular connector is almost completely embedded into the node, with no effect on its small form factor.

Figure 6.5: Output node connected to the terminal connection module, with the Blue and Red channels toggled on.

6.5 Hub

Because of the lack of size restrictions, there was much more liberty in designing the hub, however it was still important to keep the design small and portable to make operation in the field much easier. Beside the inner section that contains the PCBs, there are three main external features on the hub: the charging docks, the Bluetooth antenna, and the USB connector. Since the hub acts as a charging station and storage for the nodes, the top layer of the hub incorporates five mating docks with the same magnetic connector system used by the nodes, this enables the hub to charge the five nodes simultaneously through a mating pogo connector at each dock. The magnets also help secure the nodes to the hub during travel and storage. The hub is a wired system and needs to be plugged into the computer via a USB cable for both communications and power for charging the nodes. This is done by having a small opening which allows a Micro-USB cable to securely connect to the hub's microcontroller board. On the other side, the hub has to also communicate with the nodes in the network wirelessly. Improving range and connection quality by adding an external antenna to the nodes is not possible if we wish to maintain a small footprint, the hub, however, is a perfect candidate for an external antenna. A 2dBi dipole antenna is incorporated into the design and secured firmly on the inside of the hub by an RP-SMA screw connector, while still allowing the antenna to be rotated horizontally for easier storage. This greatly increases the range of the entire network without impeding portability and size.

Overall the mechanical designs of the hub, the nodes, and the magnetic connections allows for a small, portable, and lightweight system that could be transported and deployed anywhere, rapidly, and with minimal effort.

Figure 6.6: The completed hub and nodes. The Analog, Output, and IMU nodes can be seen with some docked and some undocked, showing both sides of the connectors. The dipole antenna is also displayed in the back.

6.6 Connection Modules

The connection modules are small passive devices that simplify the connection setup between the Analog and Output nodes and external devices. Although it is possible to solder the wires from the nodes connection cable directly to the external sensors or output devices, sometimes a more simple and temporary connection is required. The modules interface between the node connector and various different connection methods, including a terminal block for quick-connect jumpers for breadboard and other prototyping, Grove connectors for off the shelf sensor and output modules of the Grove ecosystem sold by Seeed, and a two pin connector for simple resistive sensors such as a force resistive sensor or a photoresistor sensor.

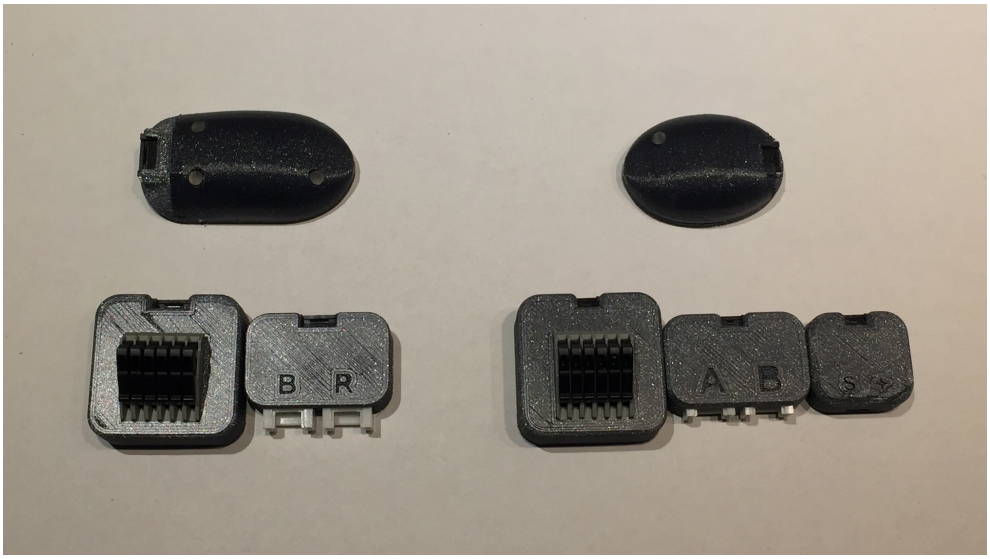


Figure 6.7: The various connection modules allow the Analog and Output nodes to easily connect to a a variety of external inputs and outputs. Left: The Output node supports a terminal block module and a two channel Grove connection module with white rectangular headers for the Blue and Red channels. Right: The Analog node also supports a similar terminal block module, a Grove connection module with two inputs A and B for the two analog channels, as well as a single resistive sensor connection module.

Chapter 7

Validation

When presenting a novel solution such as the system discussed here, it is often insufficient to simply design and build a prototype, it is also essential that the system is tested and its features validated thoroughly to ensure everything operates as expected, and fulfills the criteria it was set to meet.

Each core component and feature was evaluated to obtain the practical specifications of the system. This includes everything from the software's reliability, to the sensor network communication range, and the node's physical robustness.

7.1 Application validation

The computer application is an essential component of the system, without which the user has no contact with the sensor network. As such, each subsystem and feature was tested thoroughly after each iteration during development to ensure that errors and bugs were dealt with as they arose.

Furthermore, since the application was the only subsystem running on the user's hardware, it was essential to ensure support for as many systems as possible. To verify

this, a full installation procedure was done on multiple laptops and desktops, along with a sequential test of all system features on each machine using the following checklist.

1. Application starts with no errors.
2. Hub is detected if connected via USB.
3. Online nodes are found and connected to (green disks).
4. Click each button sequentially and ensure everything behaves as expected.
5. Starting an experiment opens the visualization window.
6. Sensor data can be seen changing in real-time in the visualization window.
7. Can preview and run command sequences.
8. Can open camera window and see live footage if cameras are connected.
9. Data files and video are saved correctly and contain uncorrupted data.
10. Clicking the reset button correctly resets the system.

This ensured that there were no compatibility or dependency issues, and that the application ran without issues regardless of the underlying hardware. However, due to limited time and resources, these compatibility tests were restricted to a small number of computers, and additional tests should be conducted for more comprehensive results.

7.2 Battery Life

Compared to other energy storage systems Lithium Polymer (LiPo) batteries have a relatively poor energy density [57], as a result, the battery's energy storage decreases dramatically with size. This, coupled with the restrictions on the nodes' dimensions, ensures that the amount of energy stored in each node is very small (40mAh), even compared to smartphones (4000mAh) and smartwatches (400mAh) [1][6].

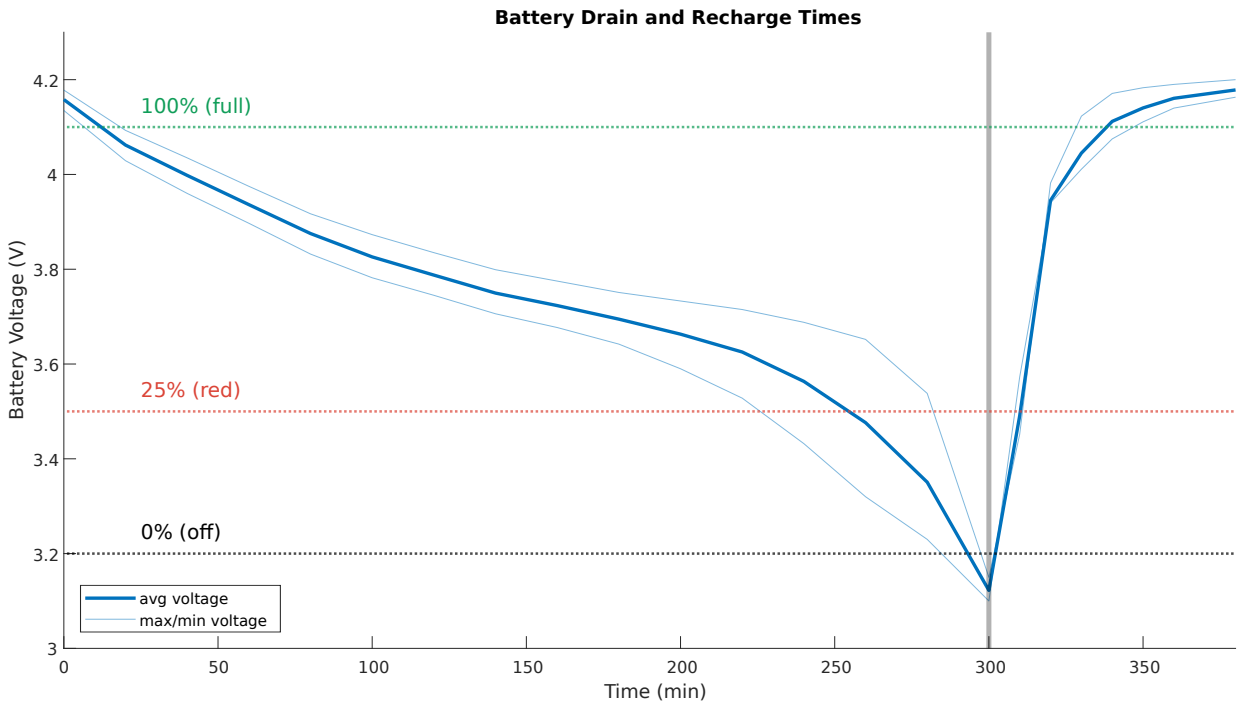


Figure 7.1: Plot of the battery discharge and charge times. The average of five charge and discharge cycles was plotted, along with the best and worst cycles. The horizontal lines represent the estimated charge percentage presented to the user in the application. The vertical gray line represents the point in time that the node was docked to the hub for recharging.

Battery life limitations can however be overcome to some degree by careful design choices when it comes to electronics and radio communication. By using a low power microcontroller running at relatively low clock speeds, coupled with a low power wireless protocol like Bluetooth, it is possible to greatly increase the runtime of the nodes by

using its available energy efficiently.

Since the current system focuses more on collecting data during experiments and less on continuous monitoring, the system should at least last for a few hours to cover the duration of an average experiment session. Multiple rounds of testing were conducted under maximal load, with continuous sensor sampling (at 250Hz) and data transmission from three IMU nodes placed 7 meters from the hub. The 40mAh LiPo batteries managed to keep the nodes running for an average of 5 hours of continuous operation. This runtime, coupled with a fast sub 1 hour average recharge duration, a side benefit of small batteries, ensures minimal downtime and daylong data collection.

7.3 Physical Robustness

As wearable devices, the nodes are in constant motion and are susceptible to being bumped into other objects or potentially falling off the body during extreme activities. This was taken into consideration when designing the node's mechanical design and and is apparent in the 2mm thick walls, the screws used to secure the node shut, as well as the shell-like exterior design.

To ensure those design decisions offer practical protection, a simple drop test was conducted. The system was setup with a single IMU node sending continuous data to the hub. The node was then dropped from a height of 2 meters while the plot was monitored for missing data. The drop test was then repeated a total of 10 times on the same node. The node survived all drops without any visible damage and continued to operate as normal.

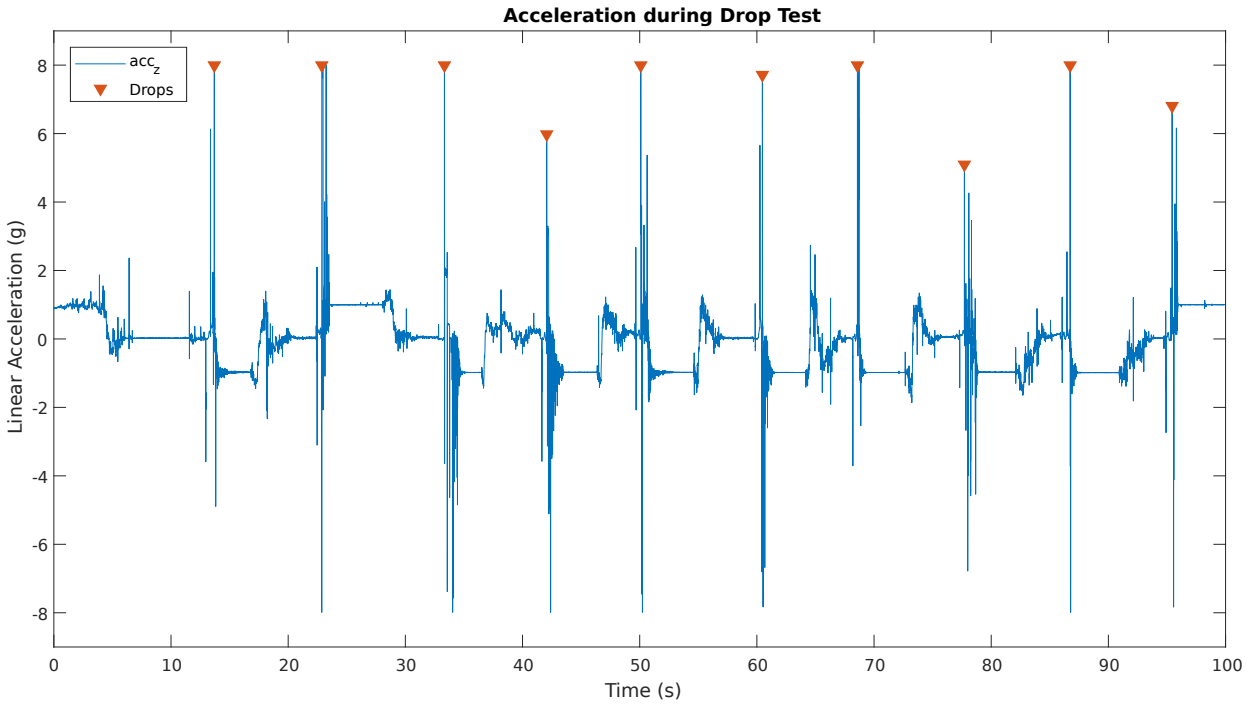


Figure 7.2: Node drop test. The acceleration along the z-axis of the node was plotted as it was dropped from 2 meters. The peaks caused by the node hitting the ground are represented by the orange triangle and were detected by a simple peak detection algorithm. The continuous plot shows that the node kept sending data throughout the drops.

7.4 Magnetic Connector

An important part of the mechanical design of the system is the magnetic connector at the bottom of each node. This connection has three core requirements: ensure the nodes do not fall off accidentally during intense activities, ensure the node is easily removeable by hand with no additional tools, and ensure good electrical contact between the node and hub pogo pins for charging.

In order to reach a good balance between the accidental and purposeful force required to detach the nodes, the magnetic connector went through multiple iterations with different magnet shapes, sizes, and number before settling on the final design.

To test the connection strength, four nodes were attached to the body, one on each

arm and each leg. The wearer then tried to shake off the nodes by moving their limbs vigorously. After a 3 minutes, all nodes remained attached, validating that under normal and even vigorous activities, the nodes have a minimal chance of falling off accidentally.

7.5 Range

An important aspect of any wireless communication system is its range. To test the system's operational range, three IMU nodes were attached to the body, one on each foot and one on the chest, the wearer was then asked to slowly walk away from the hub as the software was continuously monitored for data loss.

The test was repeated a total of ten times, five using only a chip antenna on the hub, and five with the final external dipole antenna design, in order to validate the added range provided by the external antenna. The results showed an average range of 5 meters with the chip antenna and an extended range of 15 meters when using the external antenna.

These tests showed the effective range of the system when the nodes lie within line of sight of the hub, with no objects obstructing it. The system was also tested in a gym, university labs, outdoor fields, as well as an ice rink and was found to work well as long as the wearer was within a 10 to 15 meter radius from the hub.

7.6 Synchronization

As a distributed wireless system, synchronization is essential to ensure that sensor data from different nodes are properly aligned, allowing the user to study time intervals between various events.

To test this, three IMU nodes were first aligned on a table while the sensor data

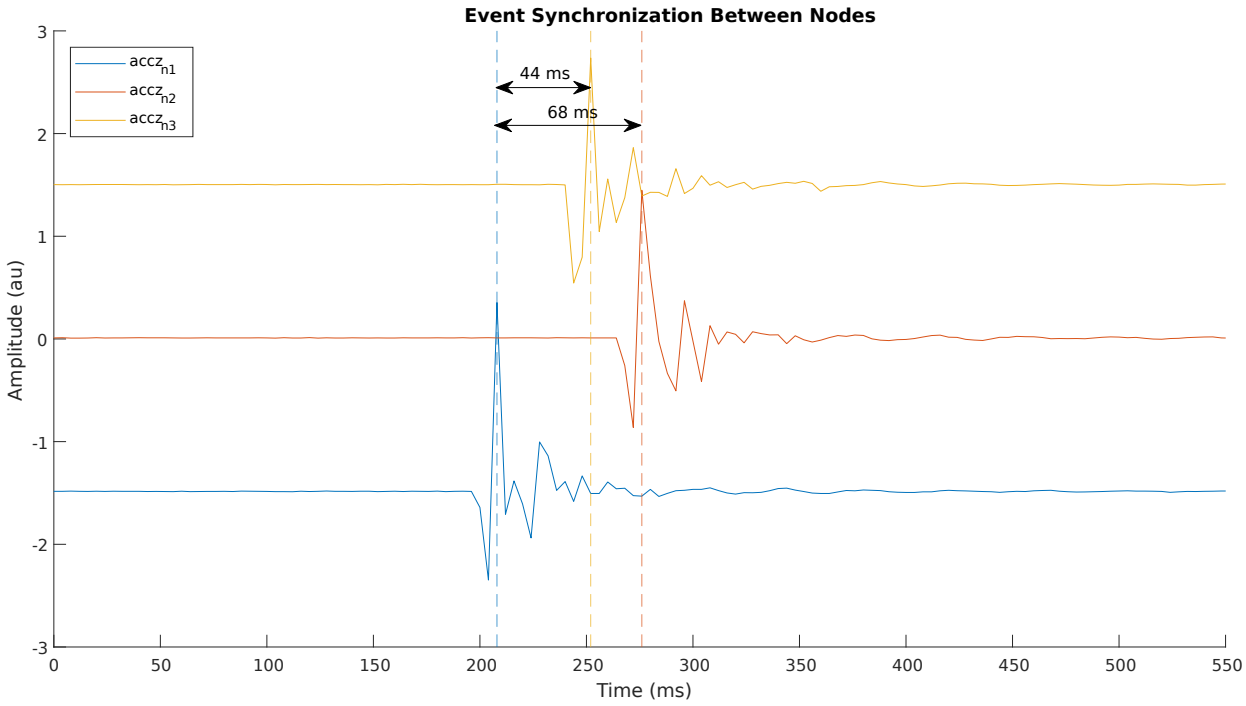


Figure 7.3: Synchronization accuracy between three nodes. The nodes have slight timing variations caused by inaccuracies in the synchronization process. This can be seen by the time differences between the acceleration peaks of the nodes caused by a single equidistant event. The synchronization accuracy can be measured by finding the duration between the peaks.

was continuously recorded. The table was then hit periodically every 5 seconds over a period of 5 minutes. This caused acceleration peaks whose timing could be then compared between the nodes as seen in Figure 7.3.

As we can observe from Figure 7.4, over the 5 minute period, no node drifted more than 260 ms away from another, while the median time shifts between nodes averaged around 82.7ms. The nodes tended to drift over time, but the buffering system, along with the 10 second resynchronization signal, ensured that all nodes remained in sync throughout the experiment's duration.

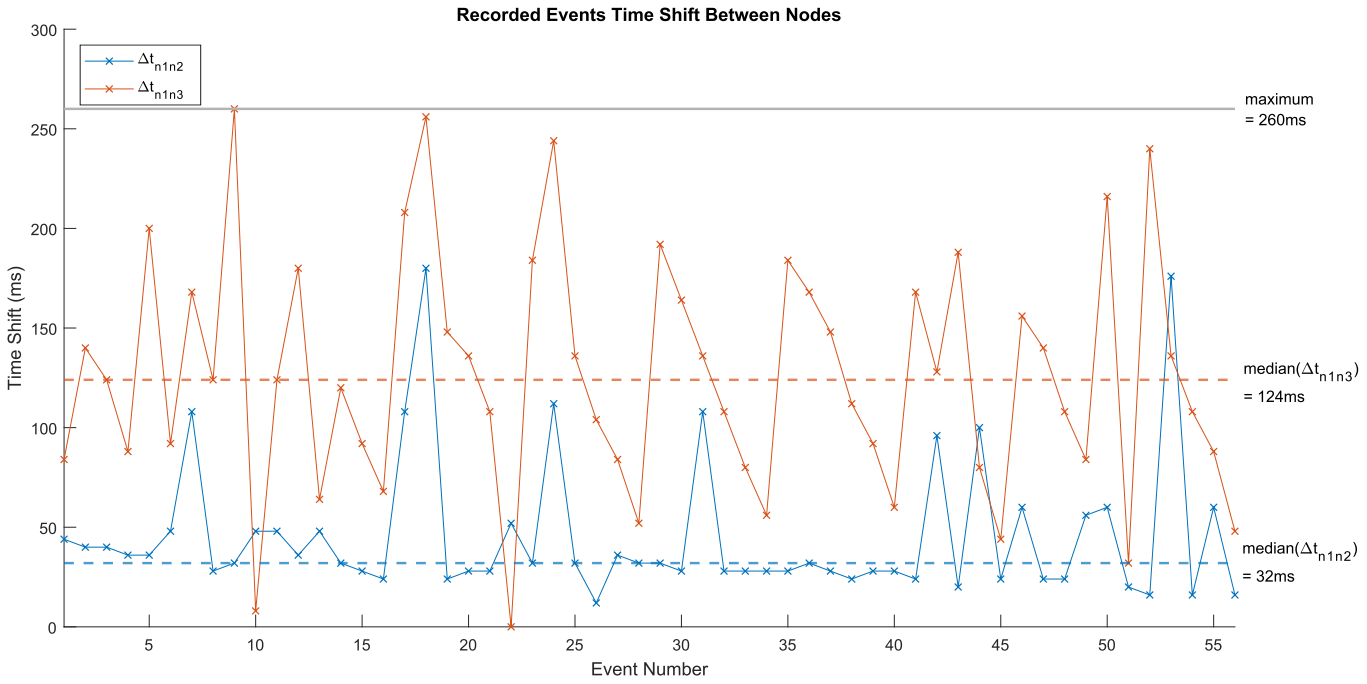


Figure 7.4: Synchronization shift over time. Plotting the time difference between the peaks over multiple events allows us to get the median and maximum synchronization accuracy between each two nodes. We notice that node 1 and 2 were more in sync than the other combinations as seen by the 32ms median shift.

7.7 Data Loss

Due to the low latency (sub 100 ms), high throughput nature of the system, data loss can occur when data from a node arrives too late, and the buffering system has to discard it to keep a consistent low latency. As an instrumentation system, data quality is of high importance and minimizing data loss is a top priority.

There are two key parameters associated with data loss, the number of concurrent devices and distance from the hub. As more high throughput devices are connected, network congestion becomes an issue [58], causing increased stress on the hub and increasing the potential of missed data packets. Similarly, increasing the distance between the nodes and the hub can cause similar issues due to late data packets and increased

susceptibility to electromagnetic interference (EMI) as a result of the low radio power output on the nodes [59].

To evaluate the effect of these parameters on data loss, an experiment was setup to collect a continuous 10 minutes of data at different distances from the hub and with a different number of connected nodes.

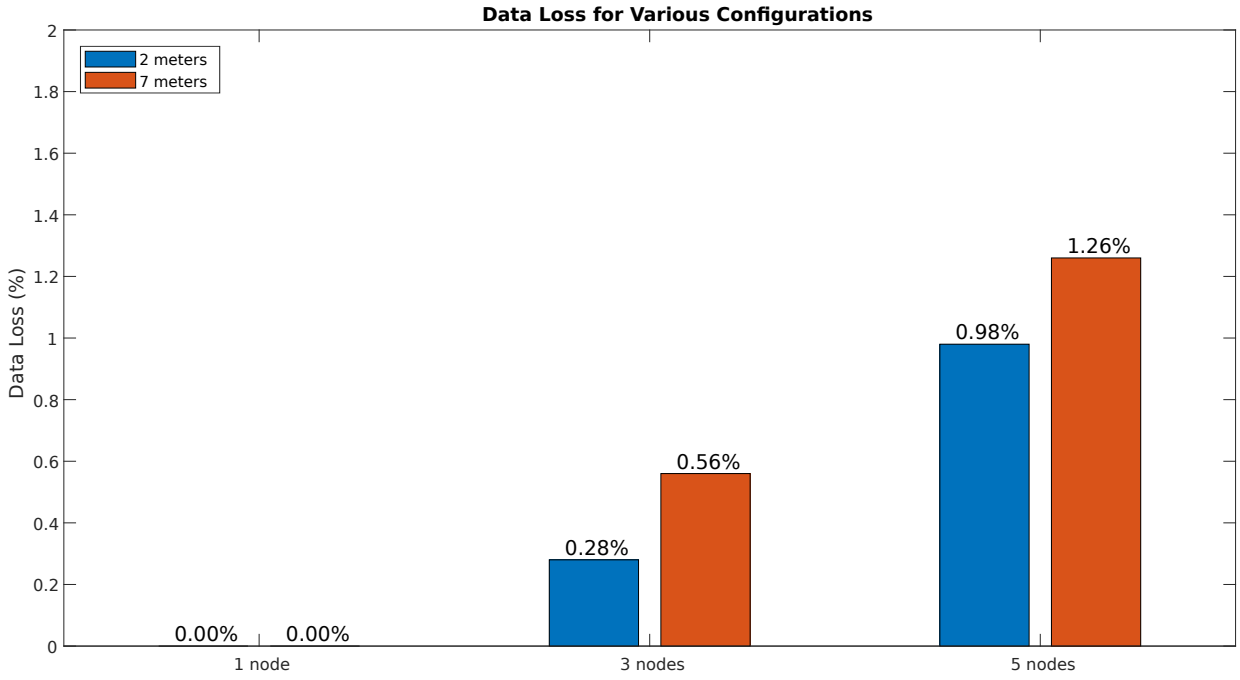


Figure 7.5: Bar plot of the data loss under different configurations. The data loss was tested under different configurations including: number of concurrent nodes represented by the brackets in the x-axis, and the distance to the hub, represented by orange for 7 meters and blue for 2 meters. The y-axis represents the percentage of sent packets lost.

As we can see from Figure 7.5, with a single connected node, we observe no data loss whatsoever, all packets are received by the hub at both 2 and 7 meter distances. As more nodes are added to the network, we observe an increase of data loss for both distances. The same can be said of increasing the distance from the hub, as the distance is increased from 2 to 7 meters more data loss is observed across the network configurations. This

is in line with our expectation that both the distance and number of connected devices are positively correlated with the amount of data loss.

However, it is important to note that even in the worst case tested, with all nodes active and 7 meters away from the hub the total number of lost data packets comprises 1.26% of all the data sent. Even though this could still be a problem for some applications involving high frequency signals, it might be less of an issue for many other applications, especially since the user is aware of the lost data and could either disregard or patch the missing segments.

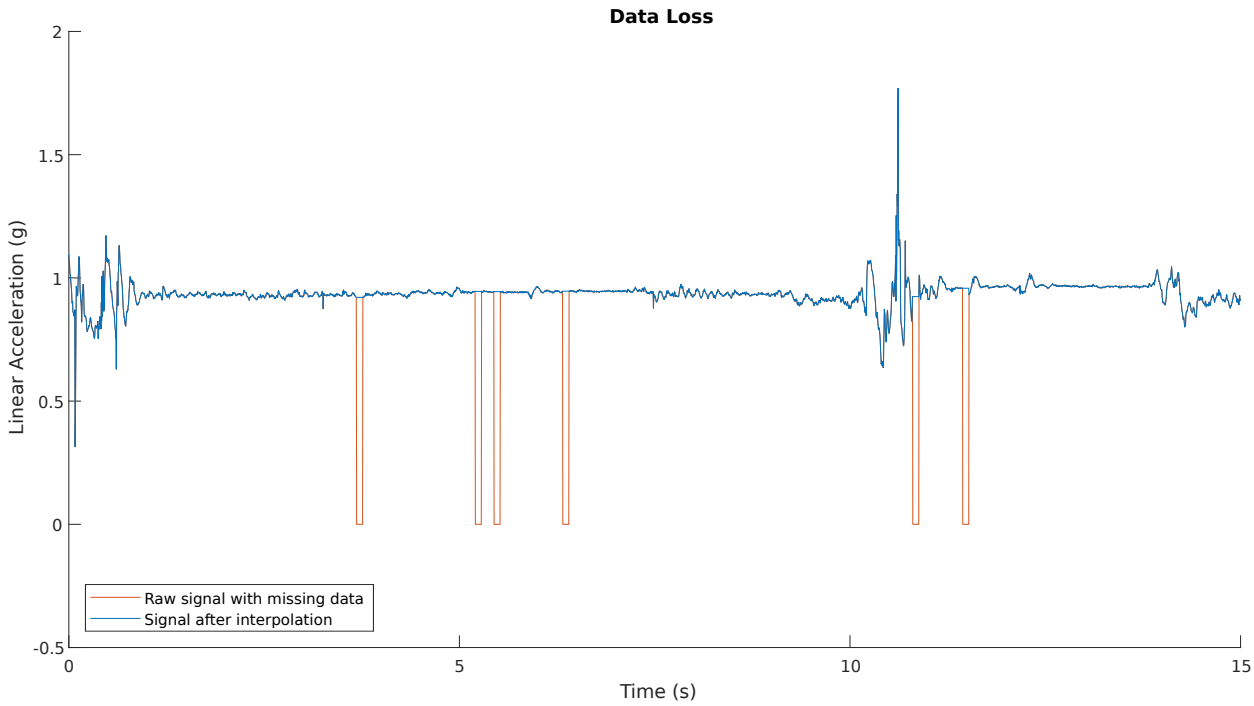


Figure 7.6: Plot of data loss fixed by interpolation. The original signal with data loss in orange can be patched by simple interpolation over the lost data, this results in a clean signal represented in blue.

In Figure 7.6, the missing segments can be patched by interpolating between the known data points. The data loss often occurs in 80ms segments which is small enough to patch the signal without loosing much information, especially for low frequency signals.

Chapter 8

Experiment

Aside from testing features individually, an experiment was designed to demonstrate the system's performance under realistic conditions, as well as to evaluate user experience and ease of use. After obtaining ethics approval (Carleton #109202), two external experimenters were chosen to run the experiment and report back on the results and information regarding the system's performance, including: any problems or bugs that occurred, desirable modifications or enhancements for future experiments, as well as a general assessment of their experience using the system.

8.1 Setup

The experimental setup involved measuring small movement and physiological changes caused by distractions during driving. A driving simulator (beam.ng [60]) with a connected USB wheel and pedal were used to simulate driving in a controlled environment, with minimal setup overhead and safety concerns.

The subject then wore three IMU nodes attached using snap anchors to the right hand, the right foot, and the forehead. The Analog node was attached to a heart rate

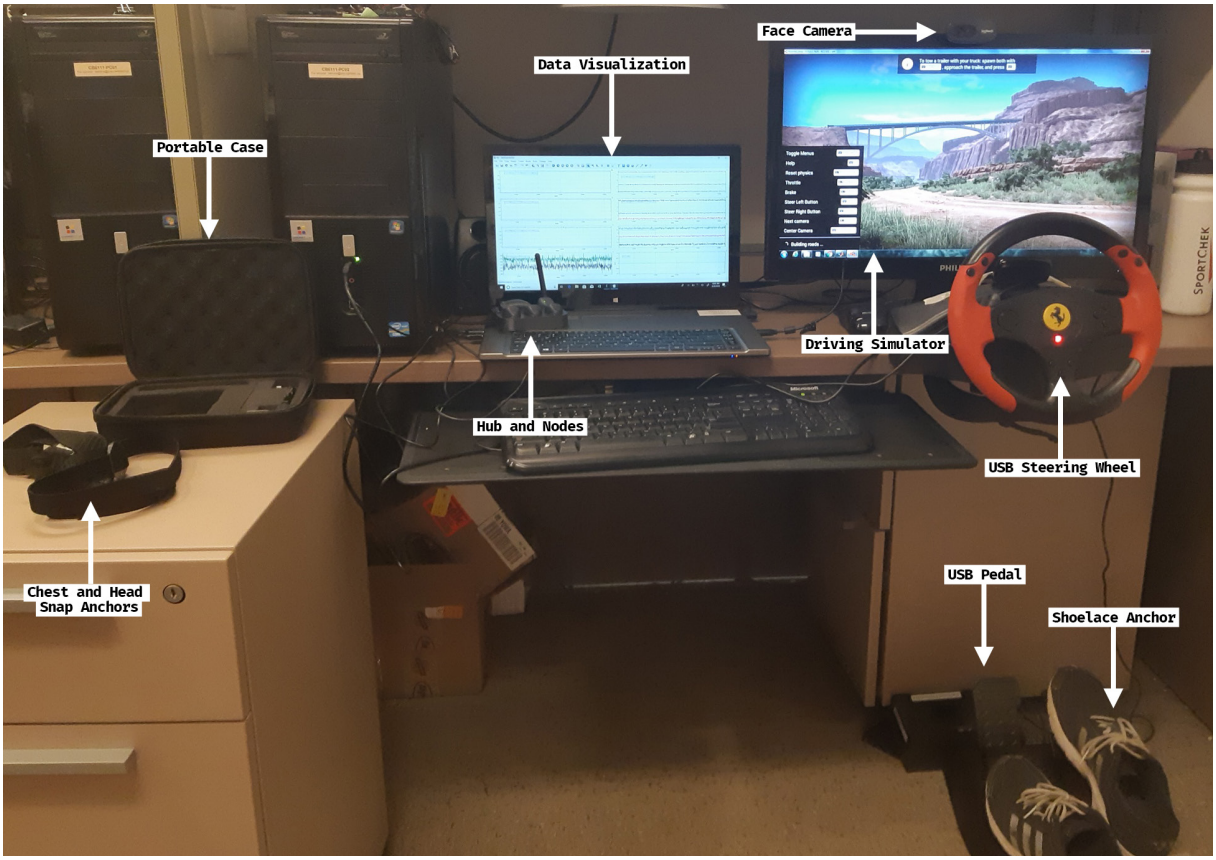


Figure 8.1: The experiment setup. The driver was instrumented using nodes attached to different snap anchors including a shoelace, head, hand anchors for the IMU nodes and a chest anchor for the Analog node with ECG module. The driver then used the beam.ng driving simulator along with a USB steering wheel and USB pedal to conduct the each trial. The face camera was attached to the monitor facing the driver. The experiment was recorded using a second laptop used by the experimenter for data collection and visualization. The custom fit portable case allowed the experimenters to comfortably pack the system for storage and transport.

sensor module and connected across the subject's chest to measure their Electrocardiogram (ECG) signal. The Output node was also used as an indicator signal for various events during the experiment. In addition to the nodes, two USB cameras were connected to the system, one facing the driver to capture facial changes and another as a monitoring camera for the whole setup.

Figure 8.2: IMU node for measuring head motion. The node was connected to an anchor with a flexible strap, forming a comfortable and easy to wear headband.

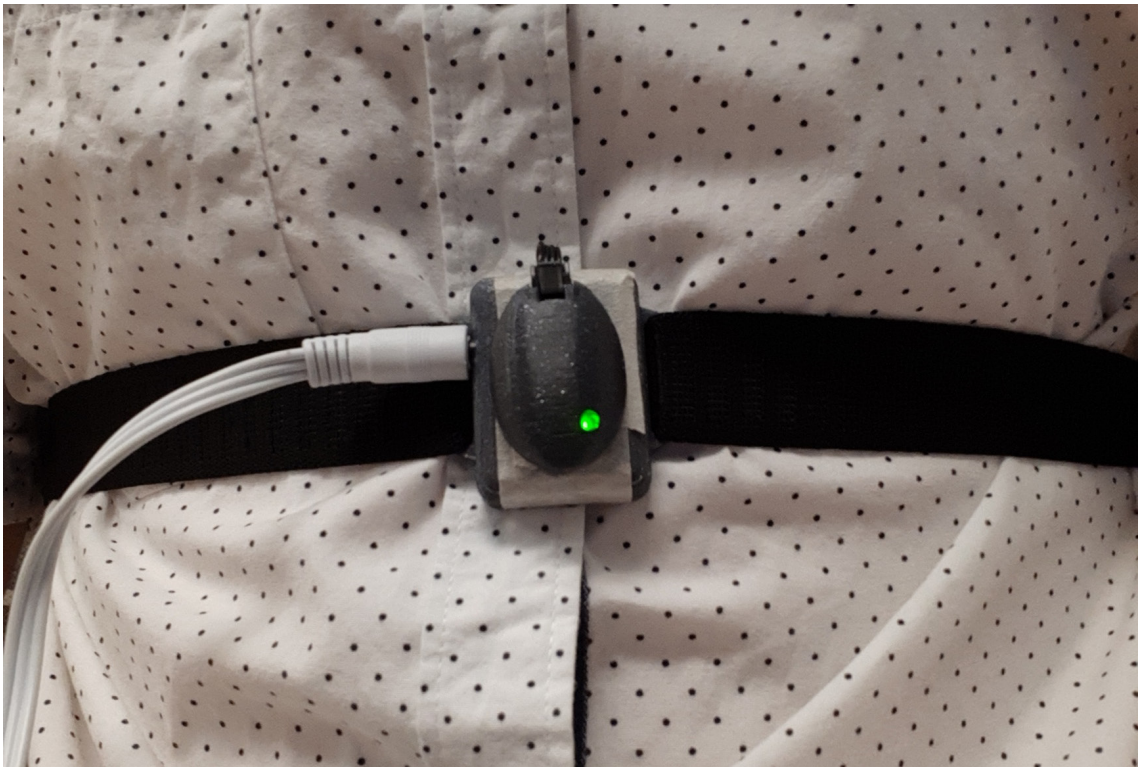


Figure 8.3: Analog node for measuring ECG signals. The Analog node was connected to a special ECG module using the same flexible strap used in the head anchor. The ECG module attaches magnetically to the node using the Magnetic Connection System, and then connects using the Analog node's external connector. The ECG module is also connected to three leads with gel electrodes on the subject's body using a standard ECG barrel connector.

Figure 8.4: IMU node for measuring foot motion. This IMU node is attached to the shoelace snap anchor, which attaches to the shoe securely through the laces. This allows the easy removal of the node while maintaining a strong connection to the shoe even under intense motion.

8.2 Procedure

The experiment was divided into two stages: a control stage with the driver following a simple route with no distractions, and a second stage in which the driver focuses on an optical illusion such as the one in Figure 8.6 for 10 seconds before initiating the drive. Five subjects were chosen by the experimenters, all males between the ages of 18 and 23 years old. Each performed five control trials followed by five distracted trials.

For the control trials, the drivers were instructed to first accelerate to 70km/h and maintain this speed until arrival to a stop sign, at which point they would stop, wait for what they perceived to be 10 seconds, look left then right, followed by making a right turn and continuing straight until instructed to stop.

For the distracted trial, the drivers were instructed to focus their attention on an optical illusion presented to them on the screen for as long as the Output node blue indicator LED was on. As soon as the LED turned off, the drivers were to start driving, following the same route as the control trial.

After recording the motion and heart rate data from the subjects, the data files were then sent to us for analysis along with the experimenter's assessment and opinions of the system.

8.3 Analysis

For most applications data collection is often only the first step of the experiment, and without great analysis, the recorded data is often meaningless. The current version of the system only offers a solution to data collection, however, the work on integrating analytical tools that would enhance the experimenter's workflow is on the future roadmap.

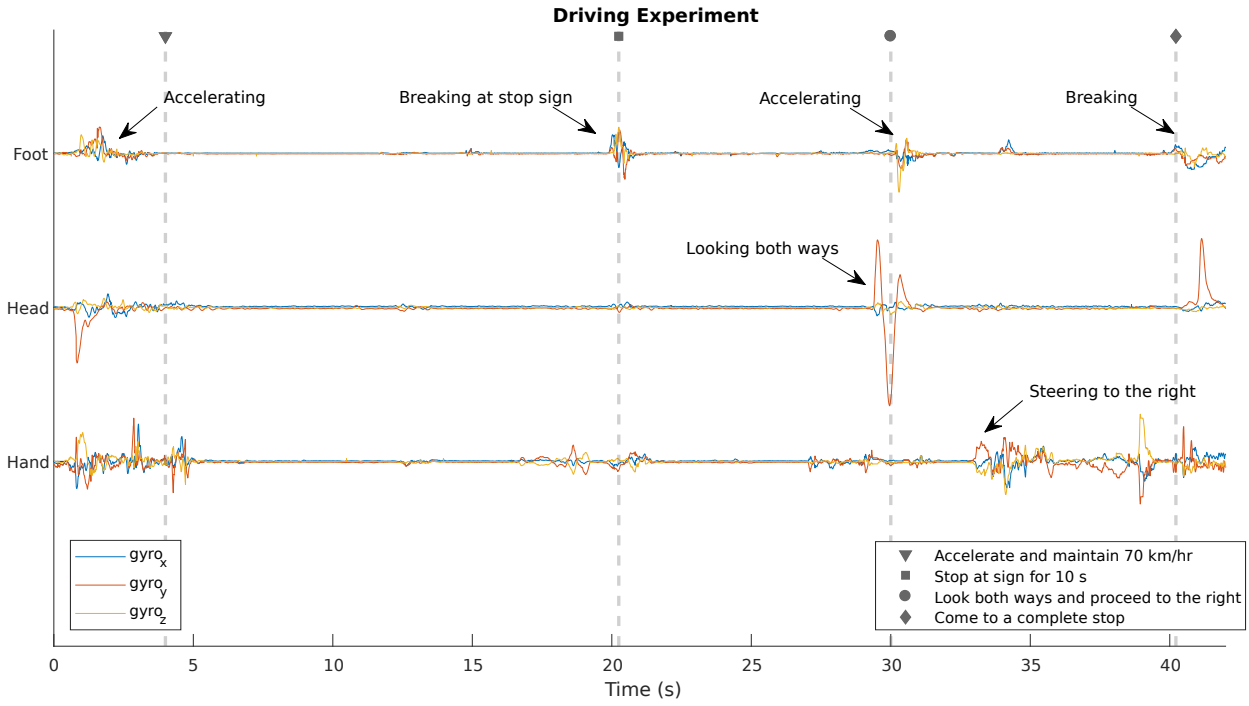


Figure 8.5: Plot of the motion experienced during the control drive. The drive is delineated by a number of events represented by the vertical gray lines. The motion of the three IMU nodes show characteristic patterns during each of those events: foot motion occurs during acceleration and breaking, head motion occurs right before making the right turn, and hand motion is most prominent when the turn is made.

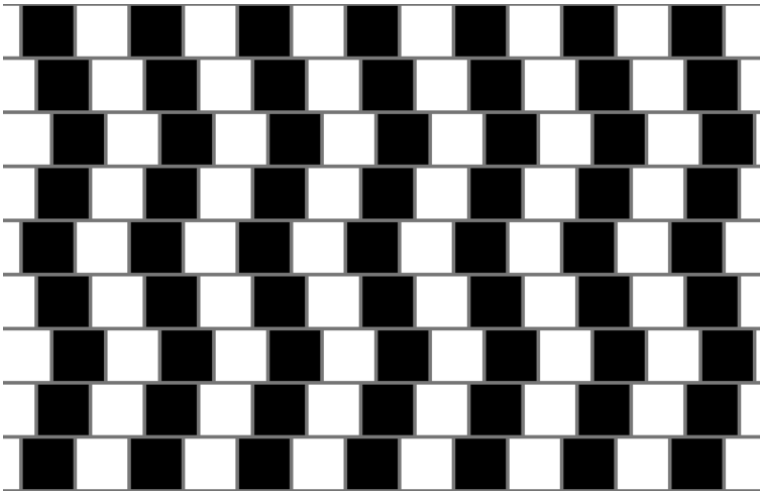


Figure 8.6: An example of the optical illusion shown to the subject at the start of the distracted trials.

To show an example of the analysis that could be done with the collected data and validate the usefulness of the system in providing real-world valuable information, we've analyzed the data from each of the nodes in MATLAB [61] to explore any possible effects the distraction could have had on the subject's driving.

8.3.1 Heart Rate

The Analog node provided a clean ECG signal from which the subject's physiological information such as heart rate (HR) could be extracted. Figure 8.7 showcases a simple peak detection algorithm which allowed us to get the timing for each individual heart beat, from which the beat to beat interval or RR interval can be extracted. This can then be converted to a instantaneous heart rate curve showing the changes in heart rate over time.

Since an elevated heart rate is often associated with stress or anxiety, one would expect the subject's heart rate to increase during the distracted trials due to increased stress. To test this hypothesis, the average heart rates for each of the subjects' control and distracted trials were calculated using the aforementioned method. To measure the

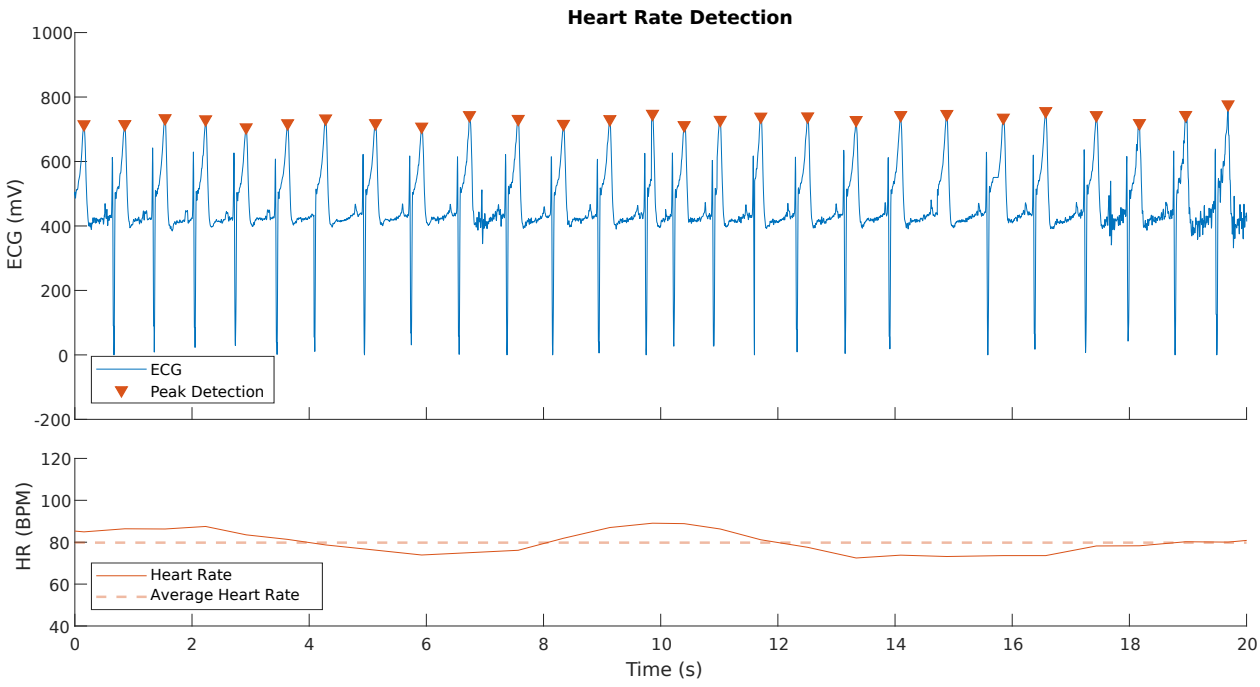


Figure 8.7: Plot of the ECG and Heart Rate during the drive. The ECG shown in blue can be converted to a HR measure by calculating the inter-beat durations after running a peak detection algorithm to get the timing of each beat.

effects of the distracted driving on each subject the percentage differences between the control and distracted heart rate averages were calculated.

As we can observe from Figure 8.8, Subjects 1 and 2 experienced a decrease in average heart rate by 2.63% and 1.68% respectively. Conversely, Subjects 4 and 5 experienced an increase in their average heart rate by 3.14% and 3.67% respectively. Subject 3's average heart rate remained constant under both conditions.

It is hard to get any clear indication of whether the distraction caused an increase in heart rate due to the contradicting nature of the results. Furthermore, the percentage changes were all very small ranging from 0.01% to 3.67%. Many more samples would have to be collected to make any informed conclusions.

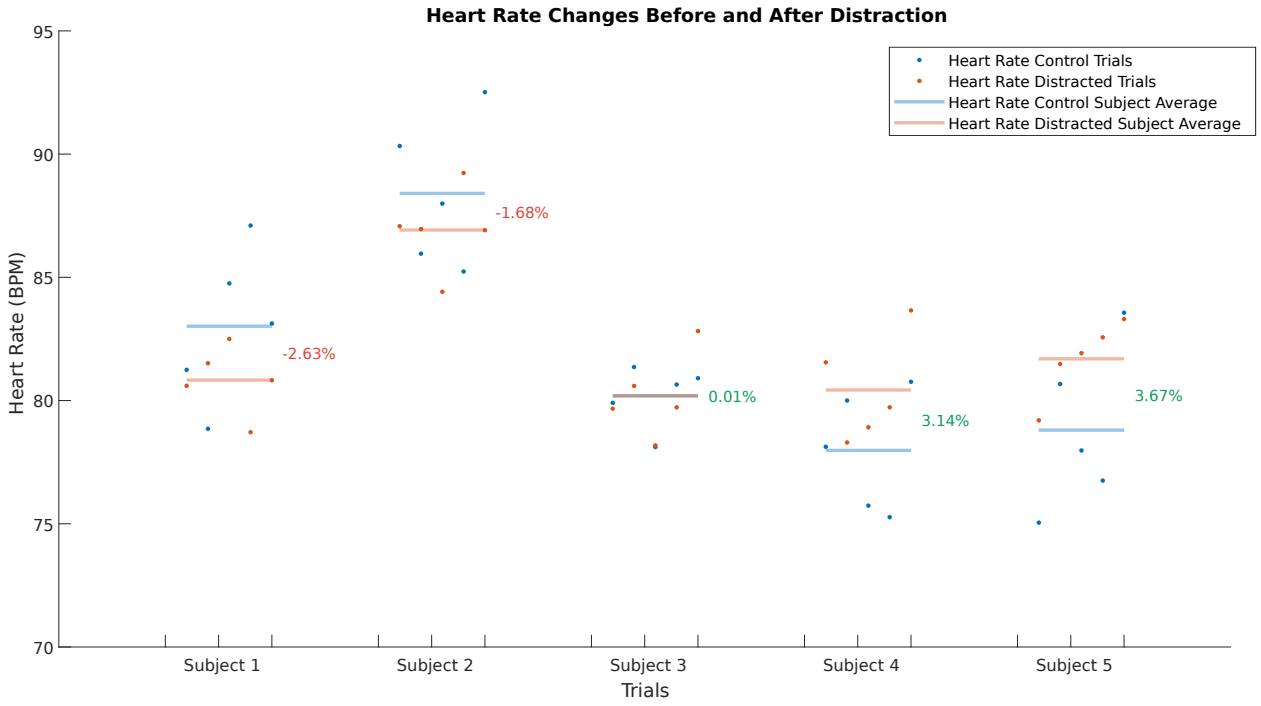


Figure 8.8: Heart rate percentage change before and after the distraction. Each dot represents the average HR for each trial, while the horizontal line indicates the combined average for each subject.

8.3.2 Head Motion

An interesting pattern captured by the IMU node attached to the subject's head occurs during the motion generated while looking both ways as seen in Figure 8.9. The gyroscope data for this pattern can be easily isolated due to the large amplitudes it produces compared to the rest of the drive where the subject's head is mostly stationary.

This head tilt is also an interesting parameter to explore any effects the distraction might have had. To do so, we compared the shape of the waveforms generated during the control trial to the ones generated during the distracted drive. The gyroscope data measured around the y-axis was chosen for exploration since it had the highest amplitudes and would thus have the most characteristic waveform. Since each trial has slight variations in the motion, each subject's trials were averaged into two characteristic

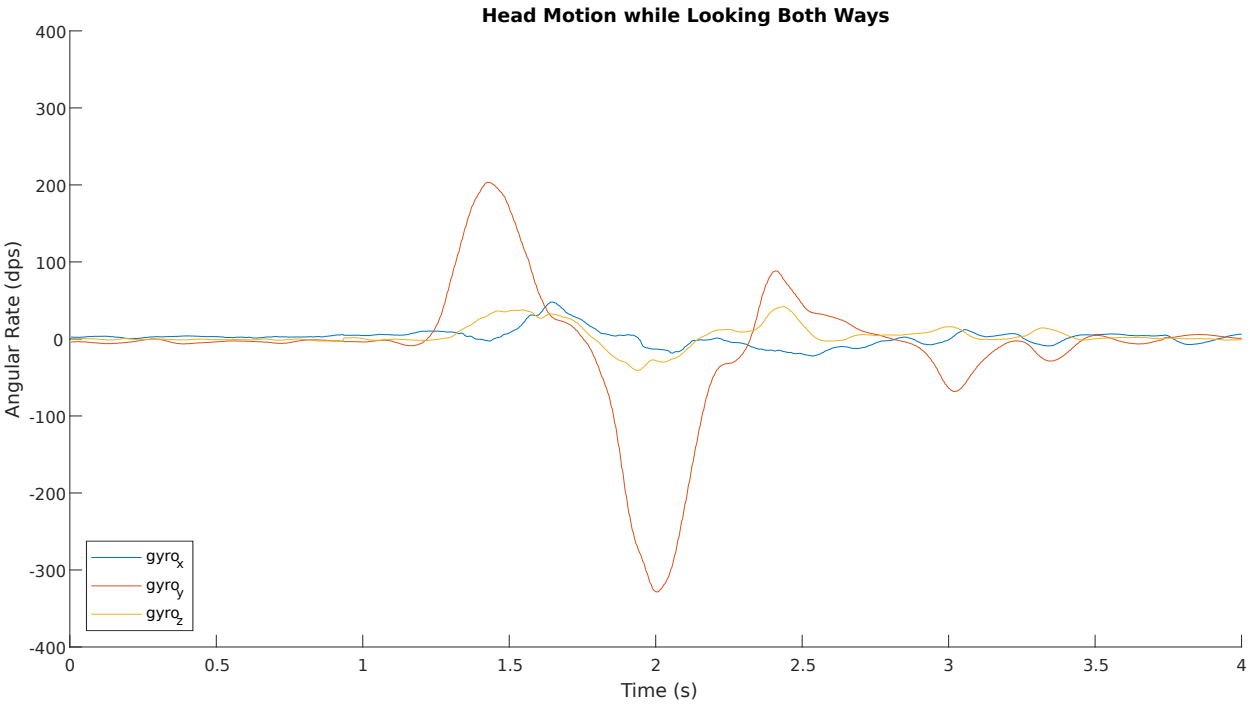


Figure 8.9: An example of the motion measured when the driver looks both ways. Since the node is placed vertically on the forehead, it is the y-axis that captures the motion from side to side. Looking left then right is represented by the peak and valley in the $gyro_y$ signal.

waveforms one for the control trials and one for the distracted trials. The waveforms were then superimposed to identify any obvious divergence between the two.

From Figure 8.10, we can see than even with a small sample size of 5 trials each, the control and distracted characteristic waveforms match very closely to one another for all 5 subjects.

In fact, when the control and distracted curves for all subjects are averaged together, the resulting waveforms show only negligible differences as seen in Figure 8.11. Based on the obtained results, it seems that the distraction had no noticeable effect on the subject's head motion while looking both ways before crossing.

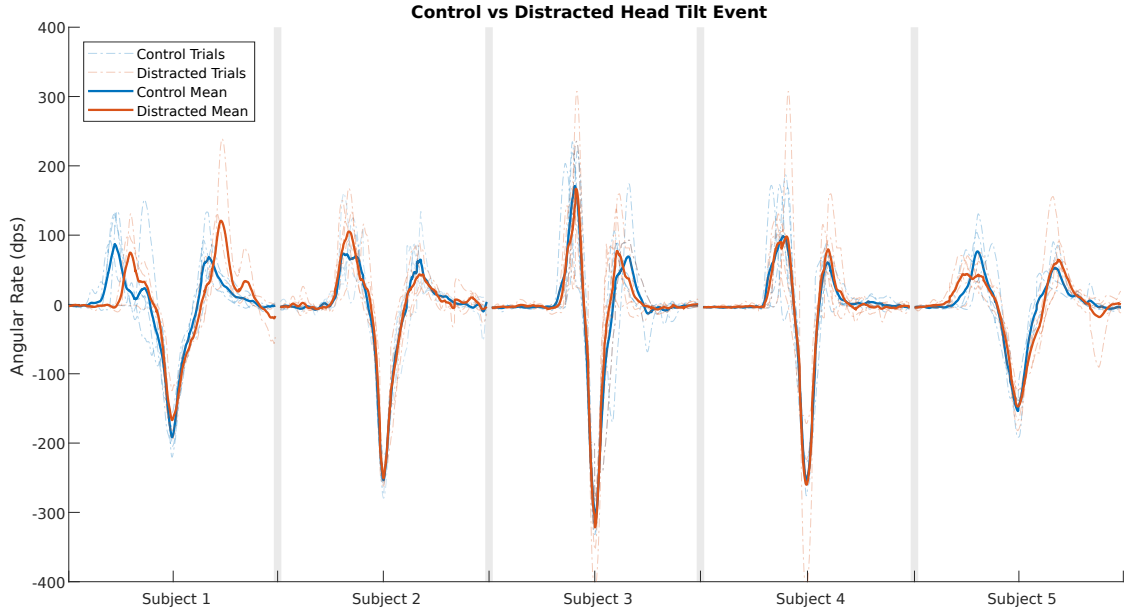


Figure 8.10: Averaged head motion $gyro_y$ curves for the control and distracted trials for all five subjects.

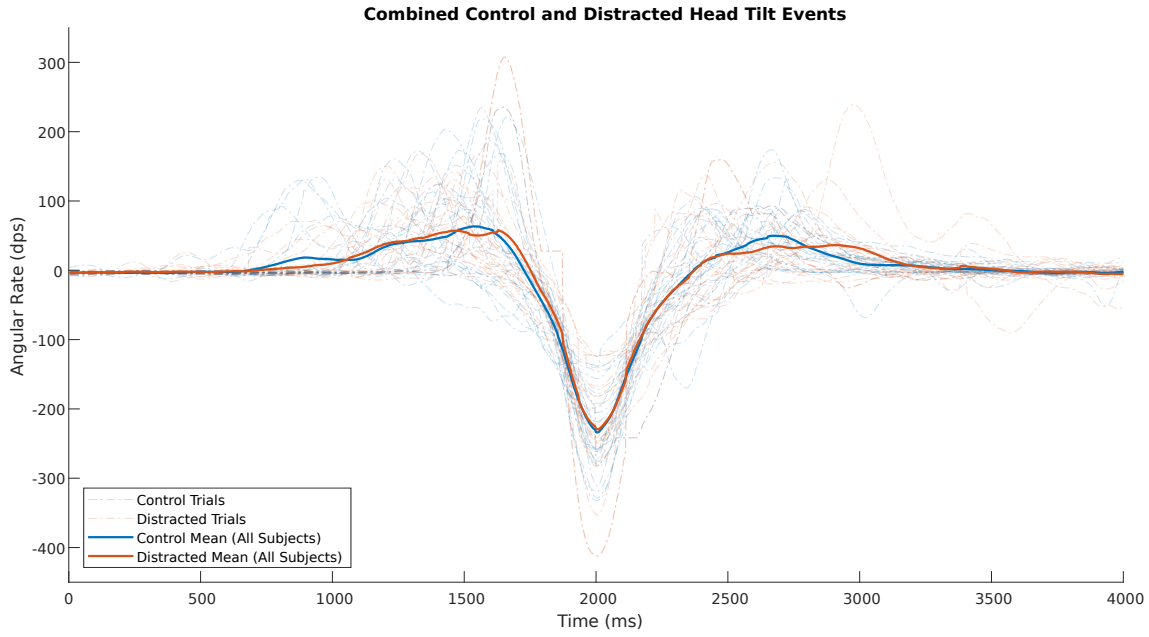


Figure 8.11: Average head motion curves across all trials. The control and distracted average curves across all subjects match almost completely, even with all the variability of the individual trials.

8.3.3 Foot Motion

During the drive, the subject is instructed to stop at a sign and wait for 10 seconds before continuing. Since the subject is not informed when the 10 seconds have passed, they have to keep track of the elapsed time internally. Such a task might be easily susceptible to distractions which could have a noticeable effect on the estimated stop duration.

The foot IMU node provided valuable information regarding the subject's use of the pedals. By measuring the duration between the gyroscope peaks corresponding to breaking and accelerating, we measured the amount of time the subject waited at the stop sign.

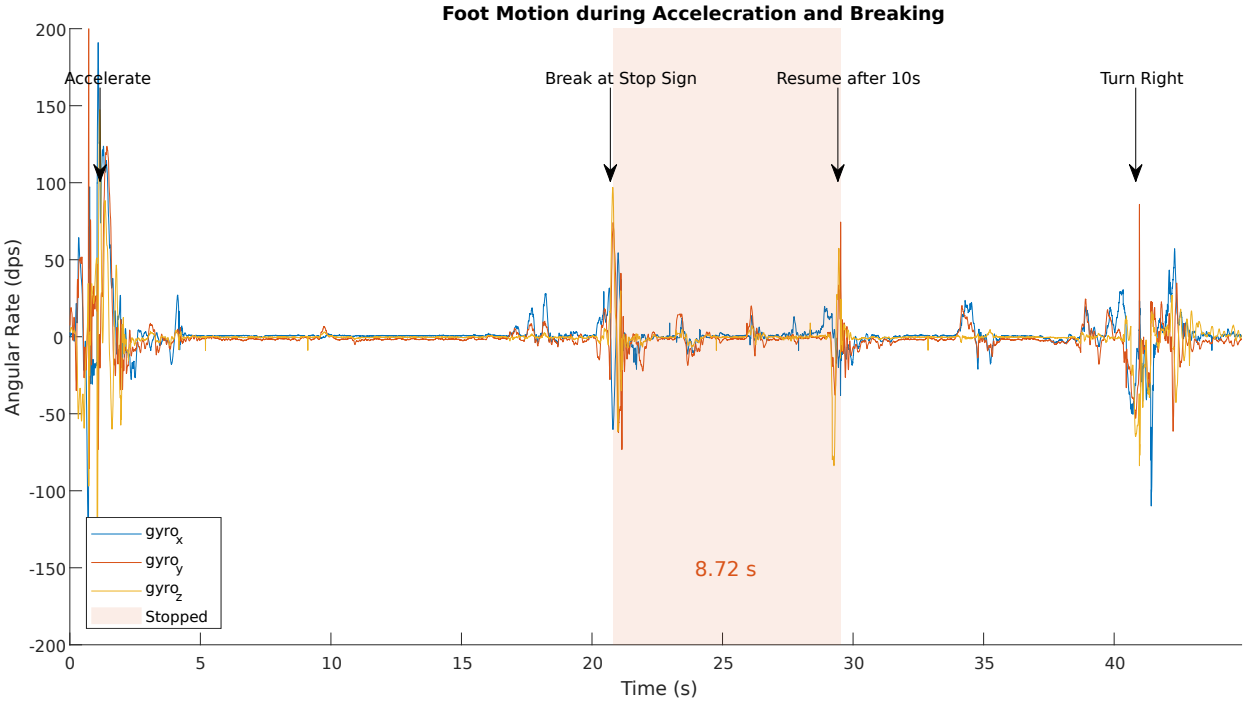


Figure 8.12: Foot motion events throughout the drive. The shaded area is the duration the subject remained at the stop, in this case 8.72s. This is calculated by using peak detection to detect the time of initial break and subsequent acceleration events.

By comparing the various trials and calculating each subject's average stop duration

as well as their average divergence from the 10 seconds mark, we measured how accurate each subject was at estimating time during their stop for both the control and distracted drives. An interval of divergence from the mark was then calculated by averaging all over- and under-shoots across the five subjects, for both the controlled and distracted trials.

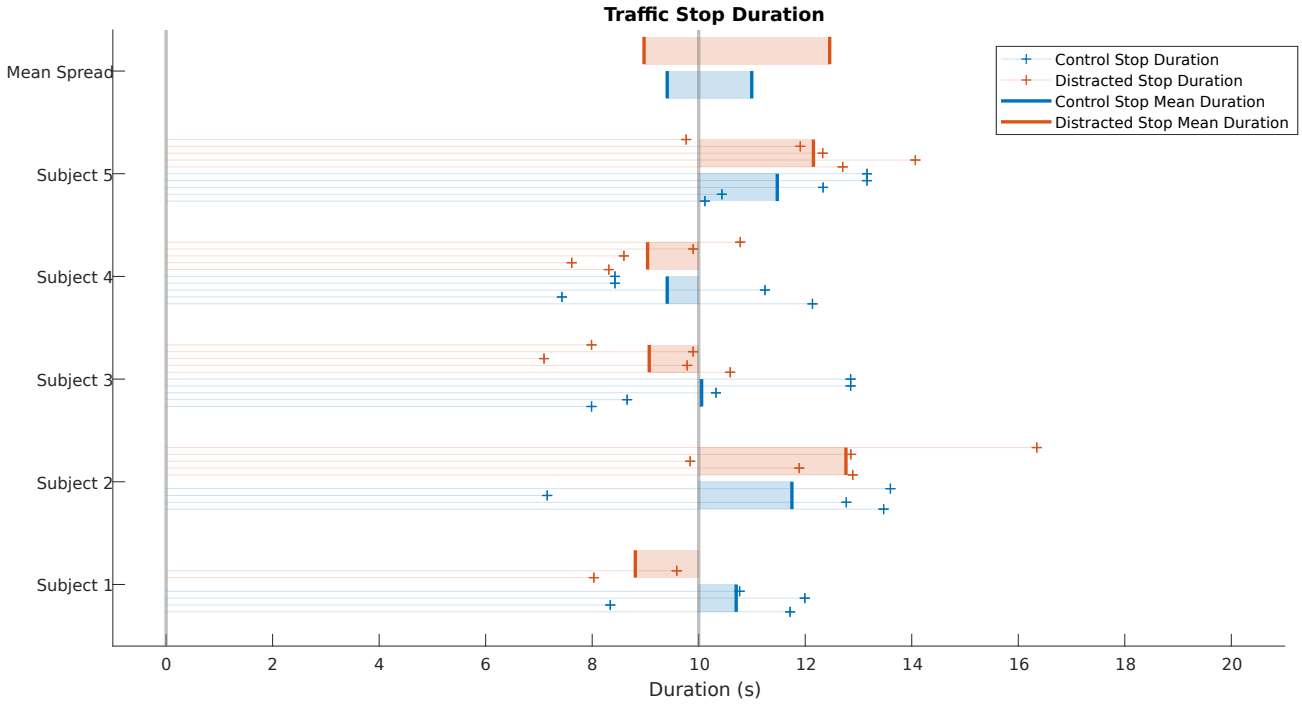


Figure 8.13: Comparison of the different stop durations for each subject. The centerline at 10 seconds represent the mark that the subjects are aiming for, a larger shaded area indicates a higher difference, and thus a lower estimation accuracy. Each trial duration is indicated by the horizontal lines, starting at 0 and ending at the + sign.

By looking at Figure 8.13, we observe that the subjects continuously overshot and undershot the mark but that on average, all five subjects had stop durations closer to 10 seconds during their control drives than they had during the distracted ones. Averaging across all trials, we see that the distracted estimates are off by 1.6 seconds while the control estimates are off by 0.91 seconds.

Even though it seems that the results consistently show that subjects are worse

at estimating their stop duration after the distraction, the sample size and differences obtained during this experiment are too small to make any conclusive remarks.

8.3.4 Hand Motion

The hand IMU node offers insight into the subject's hand motion during the drive. One characteristic motion that could be observed in the accelerometer sensor data happens when the subject turns the steering wheel to make a right turn.

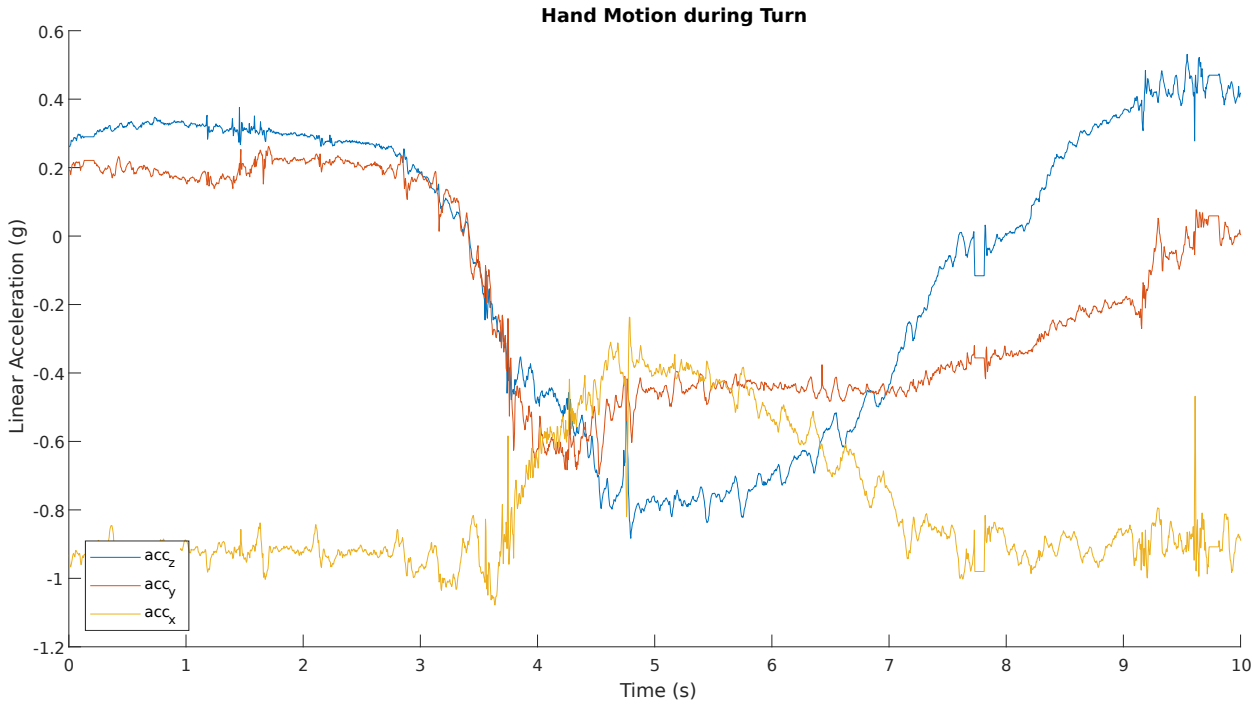


Figure 8.14: Acceleration pattern of the hand's motion during the right turn. The changing pattern is due to the change in orientation of the hand IMU node, causing the effect of gravity to be measured by different axes over time.

There are many parameters that could be extracted from this motion, such as the speed of the turn, the extent to which the steering wheel is moved, the recovery from the turn, the number of adjustments made during the turn, etc. Two of these parameters were chosen to be compared between the control and distracted trials, to find any observable changes in the subject's hand motion during the right turn.

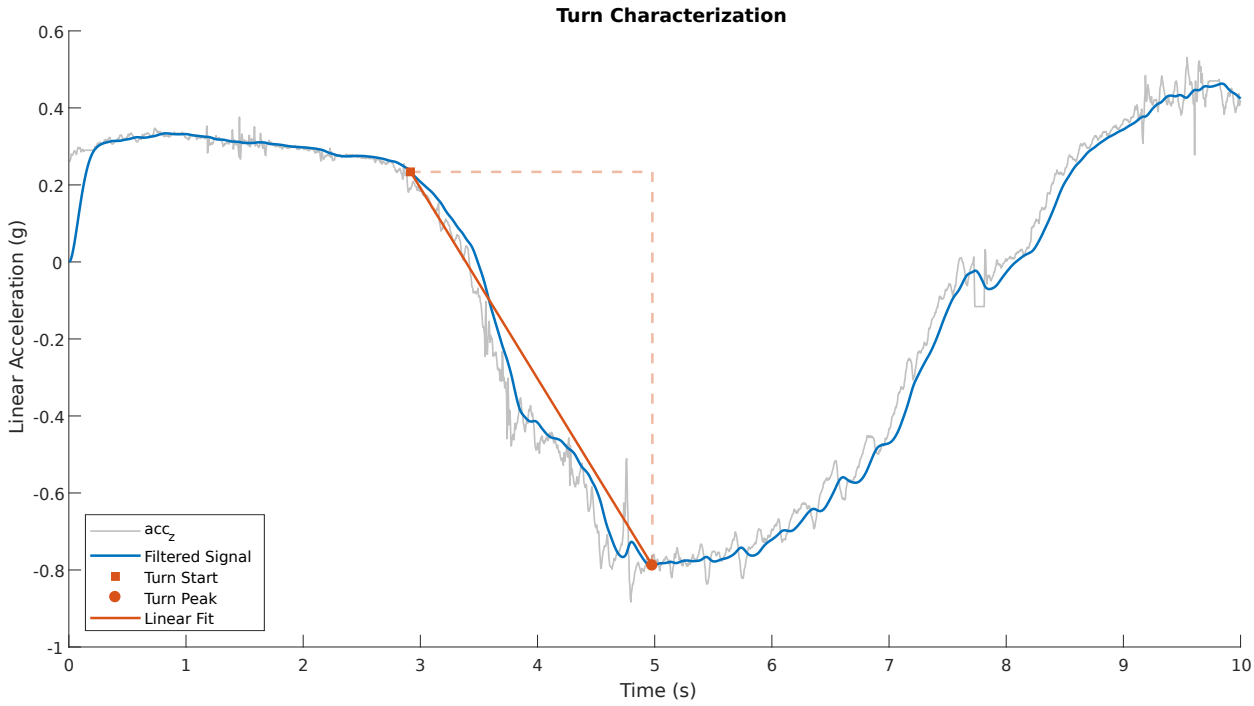


Figure 8.15: Extraction of turn parameters, including turn slope and turn peak.

The accelerometer's z-axis was chosen due to it having the largest amplitude change. Since we were interested in characterizing the turn, the high frequency components of the accelerometer signal were first removed with a 1Hz lowpass filter. The lowest point of the filtered signal was then chosen to mark the turn peak, the point at which the turn is completed and the driver starts re-centering the wheel. To characterize the speed of the turn, the slope of the line between the turn start and the turn peak is then calculated.

This procedure is then repeated for all trials, and the means of the turn slopes and turn peaks of the control and distracted trials are calculated and plotted in the scatter plot in Figure 8.16.

It is clear from the plot that the blue and orange data points corresponding to the control and distracted trials respectively have no consistency across space and cannot easily be separated. This is made even more obvious by the 2σ ellipses almost completely

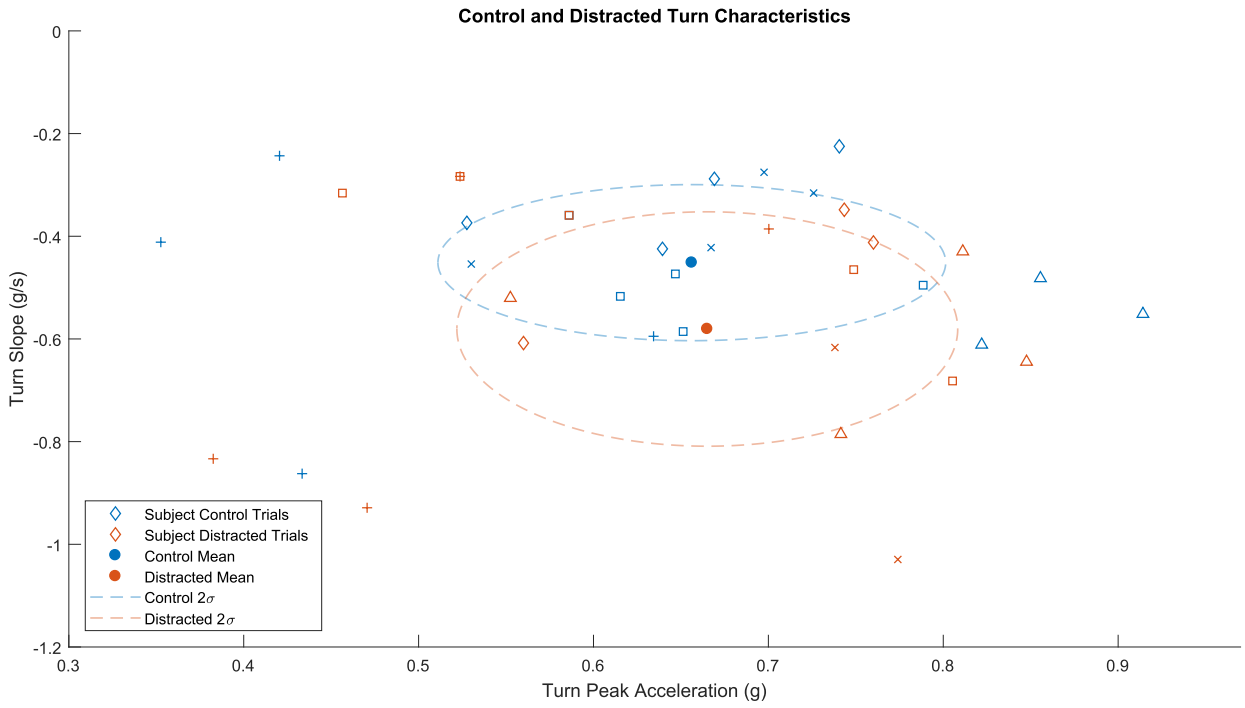


Figure 8.16: Comparison of the control and distracted trial turn characteristics. Orange points represent distracted trials and blue points represent the control trial, different subjects are represented by different shapes.

overlapping. This shows that the distraction had very little effect if any on the driver's turn characteristics.

Analyzing the data from each of the nodes provided unique insights into each subject's driving from the timing of different events to characterizing the motion during breaking, accelerating, steering and looking around, and even a proxy to stress by analyzing the changes in their heart rate. By extracting parameters from the collected data of each trial, we managed to compare each subject's control and distracted trials to explore any differences caused by the distraction. What we found, however, was that due to the negligible differences found between the two, no conclusions could be made without significantly increasing the sample size, or using more severe distractions.

8.4 Assessment

In order to obtain an independent validation of the system's functionality and user experience, the experimenter were asked to report any issues they might have encounter over the course of data collection as well as to provide their general opinion of the system and their experience using it, including its most useful features, ease of use, and any requests for changes or improvements.

The experimenters found the synchronized video capture feature to be essential to their workflow. They stated that when the cameras were off, it was very difficult to look back and analyze specific events, since the accelerometer and gyroscope waveforms were hard to decipher with no video or event logs.

The nodes were also validated to last for the entire four hours duration of the experiment session on a single charge, with 30% battery to spare.

The experimenters deemed the desktop application to be intuitive and straightforward to work with, with minimal glitches. Over a multi-day period of experiments, the application crashed a single time on startup with a "Not Responding" prompt, but was restarted without issues. Many application features were also tested and used during these experiments, including the command window for controlling the Output node, as well as the video overlay system which was found to be helpful at displaying multiple camera views in addition to the captured data in one window. In addition to the data collection stage, manipulating the recorded data files afterwards and importing them into MATLAB for analysis and plotting was also deemed easy and user friendly. Overall the experimenters found the system simple to operate while still providing many desirable features, high reliability, and a quick setup time.

These experiments have validated many of the features of the system in a practical setting. The sensor network was attached on a subject's body with IMU nodes measur-

ing limb and head movements while the Analog node connected to an external sensor module recorded physiological information. The snap anchors and magnetic connection system demonstrated their effectiveness in decreasing setup time and providing a modular wearable platform, while the distributed nature of the wireless network ensured real time data collection with minimal impact on the wearer's range of motion and comfort level. Finally, the synchronized video footage, real-time visualization, intuitive user experience, and the other features of the desktop applications were found highly useful while conducting the experiment.

Chapter 9

Conclusion

9.1 Discussion

Wearable sensor networks have elevated human instrumentation to a whole new level, allowing the continuous measurement of a multitude of parameters from motion to physiological data, without impacting the subject's comfort and range of motion. However, current solutions are limited and have missing features that could prove invaluable for many applications. For these reasons, we set out to build a modular, distributed sensor platform that improved on current designs and added much needed features to reach wider applications, with a focus on miniaturization, modularity, ease of use, and extensibility.

We built the entire system from the ground up to ensure that everything was designed to meet our requirements. These included restrictions on the size and weight of the nodes, a wireless distributed network with high throughput and sub 100 millisecond latency, a modular connection system, support for custom sensor inputs and outputs, as well as fast setup times and intuitive user experience.

The hardware for the nodes was built with a focus on size, requiring the use of the

smallest available components from a microcontroller with an integrated Bluetooth radio and antenna to a tiny 40mAh lithium polymer batter. Even the programming interface was carefully chosen to reduce the board's footprint as much as possible by ditching the standard connector in favor of flat pogo-style pads. The components were then densely packed during the layout design stage resulting 13 mm by 18 mm board which we later populated by hand soldering the components. A special board was also built for the hub, containing the five charger chips for the node's docking system.

In order to program the microcontrollers on the nodes and hub, we used the SoftDevice S132 bootloader, along with FreeRTOS and the Bluefruit52 library which we modified by adding new Bluetooth functionalities and flashed to the microcontroller using our SWD programming interface. This allowed the full support of the Arduino IDE as well as a large selection of libraries supporting various third party sensors. It is in this IDE that we later wrote all the program code necessary for the function of the sensor network, including establishing connection between the nodes and the hub, reading data from the sensors, transmitting payloads through the Bluetooth link, synchronization, communicating between the hub and the computer, and many of the other core functionalities that define the network and its operation.

To provide a more complete and user friendly experience, we built a computer application with an intuitive graphical user interface to control and operate the sensor network and to provide advanced supplementary features. The application communicates with the hub through a standard USB connection and allows the user to connect to the nodes, monitor their status, start new experiments, and visualize sensor data in real-time. The application has many other features that greatly improve the user's workflow and offer more insight into their experiment, including the ability to connect two or more synchronized camera feeds, scripting complex feedback events with the command sequence, and sending data files to the cloud.

To finalize the system, custom enclosures for the nodes and hub were designed in the Fusion360 CAD software and manufactured in-house using a 3D printer. The node's enclosure was designed to fit the PCB and battery while minimizing the overall form factor, it also included a lightguide for the node's indicator LED as well as a custom connection system at the bottom of the node. This magnetic connection system used a pogo-style four pin connector along with three neodymium magnets and is an important aspect of the node's modularity. It allows the node to attach to various snap anchor points worn on the body, making switching the nodes between individuals or changing their configuration fast and simple. This connection also doubles as a docking interface to the hub for charging and storage, making the whole system compact and portable for quick setup on the field.

The end product is a complete sensor network platform that achieved all the desired objectives we set out to meet. It manages to provide interesting new features while still fulfilling the necessary criteria for a highly versatile research platform.

- **Distributed:** a wireless system with multiple independent sensor nodes.
- **Modular:** a versatile magnetic connection system with custom snap anchor points that can be placed anywhere on the body.
- **Extendable:** input and output nodes allow the use of custom analog sensors and precise control of external digital devices.
- **Small and lightweight:** tiny form factor occupying 35 x 25 x 15mm and weighing 7 grams.
- **Easy to setup and modify on the spot:** a portable system with a docking functionality for node storage and transport with fast setup times and swapping thanks to the magnetic connection system.

- **Raw sensor data:** a comprehensive raw data storage functionality with CVS and TXT options, headers, timestamps, and filename time stamping.
- **Sub 100 ms latency for real-time communication:** a high throughput low-latency communication scheme with node synchronization, real-time visualization, and trigger detection
- **Simple and intuitive:** a user friendly graphical interface, simple one-button operation and automated installer and updater.

Reliability is of high importance for any research tool. To evaluate the system's performance and discover any potential shortcomings, we conducted a number of validation experiments on some of the system's most essential characteristics. These included node charge and discharge times, its physical robustness and magnetic connector's resilience, as well as network related tests for synchronization accuracy, system range and data loss. The computer application was also thoroughly tested on multiple devices to ensure all major bugs were dealt with. Overall, the validation results allowed us to better understand the specifications and limits of the system. Runtime and range were found to be adequate for many indoor applications with an average range of 15 meters and battery life of around 5 hours. Synchronization was accurate to below 300 ms while data loss was minimal at 1.26% in worst case conditions.

Testing the system in a controlled environment is not enough to prove its value. To further assess its usefulness in a real-world scenario, the system was handed to third party experimenters who set out to collect various sensor data from subjects operating a driving simulator. The IMU nodes were used to obtain motion information from the driver's head, foot, and hand, while the Analog node was used to capture the subject's ECG signal. This experiment was also an opportunity to obtain independent validation of the system's user experience, ease of use, and reliability. It also helped identify the

system's most valued features and desired improvements in the experimenter's eyes. Overall, the experimenters were pleased with the system, praising the synchronized camera feature, the system's ease of use, simple setup procedure, and lack of issues.

Along with their assessment, the experimenters sent us the data they had collected during their experiment for further analysis. Even though the current system is mainly focused on data collection, analysis is an essential step to convert the captured data into useable information. To give an example of the possible analysis that could be done on the data recorded by the system, we explored the data from each of the nodes using various methods to extract interesting driver behaviours and parameters such as heart rate, head tilt characteristics, steering speed and extent, acceleration and breaking patterns, among others.

By building the system with specific features and design criteria in mind, we managed to create an easy to use, effective data collection platform, which was then thoroughly validated by testing it internally and through third party experimenters. The system worked well and achieved most of what it was expected to. However, during development, we faced some challenges that were hard to overcome, some of which remained as limitations on the system, offering plenty of room for future work and improvements.

9.2 Challenges

Throughout development, we faced many challenges both expected and otherwise. Some limitations were apparent during early design stages, while others became clearer during development and were confirmed during validation.

9.2.1 Bluetooth

One integral component of the sensor network is the wireless protocol used to communicate between the nodes and the hub. Early on, Bluetooth was picked as the protocol of choice as it exhibited many features that were complementary to the low power and miniaturized nature of a distributed system. But it was also clear that Bluetooth was not without its shortcomings, having a relatively low throughput rate and range. However, with the new Bluetooth 5 introducing higher throughputs of up to 2Mbps [62], there were hopes that this increase would be sufficient to support the sensor data streamed from the nodes.

In practice however, it became clear that Bluetooth 5 had many built-in limitations that would greatly impede the potential of the system, especially when it came to latency. One such limitation is the so called Bluetooth Connection Interval which enforces a 7ms minimum duration required to complete a radio transmission [62][48]. This limited the amount of supported nodes working simultaneously to four input and one output node, as the hub needed to service all nodes on the network within a small time window before new samples came in. With new data sent every 80ms and two 7ms Connection Intervals required to service each node, the network was thus limited to five concurrent nodes. This in turn greatly limited the amount of data sent every interval, which decreased the maximum sensor sampling rate allowed. Such latency also introduces discrepancies during synchronization, as the hub has to wait a significant amount of time between

sending the synchronization signal to each of the nodes.

Additionally, in the case of any network interference which could cause the data to be sent late, packets that fail to be read while the hub is busy servicing another node would then have to be retransmitted, causing even more network congestion, synchronization drift, and potential data loss as observed during validation. The tight timing windows caused by the minimum Connection Interval, coupled with the inability to disable retransmissions [62][48], greatly hinder the ability of Bluetooth 5 to handle high-throughput low-latency applications.

9.2.2 Range

Another networking limitation of the system is its range. Many system parameters can impact range such as transmit power, receive sensitivity, and antenna configuration. Compared to other wireless technologies such as Wi-Fi or LoRa, Bluetooth has a significant range disadvantage due to its focus on size constrained, low power, applications [62].

Connecting a large external dipole antenna on the hub significantly improved the range, however, the low power nature of the node remains a limiting factor. The maximum permissible transmission power output of the node is 4dBm [32], this limited output has a direct impact on the distance the signal can traverse, following the inverse square law.

Increasing the size of the transmitting antenna on the node could help alleviate this problem, but due to the size restrictions, doing so was not an option. The node uses an on-chip antenna integrated into the microcontroller's IC package [30]. This is suitable for constrained designs as it has the smallest footprint of any kind of antenna, but results in a significantly impeded range.

A related issue is the loss of data that sometimes occurs when the wearer's body

is occluding the node from the hub [63]. This is another drawback of the node's small antenna and low power, which could result in the transmitted signal dissipating entirely into the body before reaching the hub.

9.2.3 Battery

The size of the node was also the primary factor guiding our choice of battery. Since the size of such lithium polymer batteries corresponds directly to the amount of energy stored [57], it was clear that this choice would severely limit the system's battery life. Even though a running time of 5 hours is enough for many short experiments, this limits the system's potential by excluding long term experiments and continuous monitoring applications. There are many hardware and software improvements that could lead to a much more efficient use of the available energy. For the initial version however, the battery life was deemed sufficient and not much effort was spent on increasing it.

9.3 Future Work

Before the system could be used in any practical setting, some reliability issues have to be addressed. The occasional data loss is the most serious contender as it has the potential to impede data collection. In contrast, all other limitations such as range and number of connected devices do not hinder the system's operation, but rather simply restricts it to specific environments and applications.

9.3.1 Better Radio Protocol

Since data loss issues appear to be caused primarily by network congestion, solutions that involve increasing throughput and reducing latency could help alleviate the problem. One such solution is to move to older Bluetooth protocols such as the Human Interface Device Profile (HID) used for wireless mice and joysticks which might offer a more suitable option with higher throughput and lower latency. Yet another option would be to move away from Bluetooth entirely towards a custom radio protocol, purpose-built to best suit the system's needs. Moving to another radio communication protocol could also help deal with some of the other system limitations by introducing, better range, higher sensor sampling rates, and an increased number of concurrent nodes.

9.3.2 Increased Runtime

In addition to fixing the issues of the current design, one could look to future improvements that could help deal with many of the current system's limitations. One such improvement is increasing battery life. Many aspects of both the hardware and software design could be modified to increase battery life. Shortly after the first systems were completed, a new denser battery was released with 50% more stored energy and similar dimensions, using it would greatly increase the battery life of the nodes without

modifying their form factor. There is also much to be done when it comes to reducing power consumption by increasing the efficiency of the software running on the node. Since the radio accounts for the majority of power consumption in the node, increasing the efficiency of communication between the nodes and the hub should greatly impact battery life. Reducing power consumption could also be achieved by decreasing the microcontroller's clock speed. It is also possible to present the user with an option to put the system in a low power mode for less dynamic applications, decreasing the node's reporting rate while increasing uptime.

9.3.3 Increased Range

Another clear candidate for improvement is the system's range. Using an even larger antenna on the hub can improve range incrementally, however there comes a point where even the hub could become unwieldy. A more effective solution would be to improve the node's transmission.

Even though the current on-chip antenna used by the nodes is is quite limited, there are other potential improvements to this antenna design that have the possibility to improve range without significantly increasing the node's size or changing its form factor. One such solution is to use a slightly larger surface mounted ceramic antenna or a PCB trace antenna, another is to go for a flexible antenna attached around the inside of the node's 3D printed shell.

Increasing the node's transmission power improves the system's range at the expense of higher power consumption, this would however involve using a new microcontroller and radio as the NRF52832 of the current system is operating at its maximum output power of 4dBm [32].

9.3.4 Additional Sensor Modules

The Analog node greatly increases the versatility of the system by allowing the user to integrate any external analog sensor into the network, enabling highly customized experimental setups. However, in contrast to the completely integrated IMU node, the Analog node comes at the cost of an additional external connector, cable, and sensor module. More integrated sensor nodes could be manufactured based on the same principle as the IMU node with on board sensors, broadening the user's options by having sensor specific nodes while maintaining the same compact wearable form factor.

The external custom sensor option could also be enhanced by using the node's bottom connection pins and magnetic connection system, currently only used for charging, as a communication interface for more advanced sensor modules. This removes the need for an external cable and could allow the node to communicate with sensor modules over I2C or even UART.

9.3.5 Further Validation

Additional validation procedures and tests could also be conducted to further characterize the system's behavior under various conditions.

Stress testing methods such as highly accelerated life tests (HALT) can showcase the system's reliability in reaction to shocks and vibrations, as well as temperature and humidity extremes. The reliability of the system in harsh environments and its ability to withstand dust, rain and cold are vital for many outdoor applications.

Battery life and power consumption could also be measured much more accurately using power profilers, to both identify inefficiencies in the software as well as provide better runtime estimates under various loads.

For some of the biosignal sensors such as the ECG module, additional in depth

validation of both accuracy and electrical safety is necessary to meet patient safety standards and prove the system's potential in health and medical applications.

9.3.6 Next Step

Work has already started on the next iteration of the system, and aside from the improvements to the design listed here, there were also many lessons learned building the current system that would greatly simplify and accelerate the next iteration's development stage. The new design would improve on the current one by having higher reliability, longer range, more concurrent nodes, and more sensor modules while also reducing the cost of production and streamlining assembly, resulting in a more robust wireless sensor platform with even better versatility and broader applications.

References

[1] Samsung, *Specifications — Samsung Galaxy S10e, S10 & S10+*, en. [Online]. Available: https://www.samsung.com/global/galaxy/galaxy-s10/specs/.

[2] B. L. Risteska Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," en, *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, Jan. 2017, ISSN: 09596526. DOI: 10.1016/j.jclepro.2016.10.006. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S095965261631589X.

[3] M. Alaa, A. Zaidan, B. Zaidan, M. Talal, and M. Kiah, "A review of smart home applications based on Internet of Things," en, *Journal of Network and Computer Applications*, vol. 97, pp. 48–65, Nov. 2017, ISSN: 10848045. DOI: 10.1016/j.jnca.2017.08.017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1084804517302801.

[4] B. M. Booth, K. Mundnich, T. Feng, A. Nadarajan, T. H. Falk, J. L. Villatte, E. Ferrara, and S. Narayanan, "Multimodal Human and Environmental Sensing for Longitudinal Behavioral Studies in Naturalistic Settings: Framework for Sensor Selection, Deployment, and Management," en, *Journal of Medical Internet Research*, vol. 21, no. 8, e12832, Aug. 2019, ISSN: 1438-8871. DOI: 10.2196/12832. [Online]. Available: http://www.jmir.org/2019/8/e12832/.

[5] Rubin Peter, "How Fitbit Started the Wearables Craze and Got Us All Moving," en, *Wired*, ISSN: 1059-1028. [Online]. Available: https://www.wired.com/story/how-fitbit-got-us-all-moving/.

[6] Apple, *Apple Watch Series 4 - Technical Specifications*. [Online]. Available: https://support.apple.com/kb/SP778?locale=en_US.

[7] P. Kumari, L. Mathew, and P. Syal, "Increasing trend of wearables and multimodal interface for human activity monitoring: A review," en, *Biosensors and Bioelectronics*, vol. 90, pp. 298–307, Apr. 2017, ISSN: 09565663. DOI: 10.1016/j.bios.2016.12.001. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0956566316312295.

[8] V. Savov, *Samsung's Galaxy Gear is a smartwatch like no other*, en, Sep. 2013. [Online]. Available: https://www.theverge.com/2013/9/4/4692824/samsung-galaxy-gear-features-specs-release-date-price.

[9] T. Hayley, *Everything you need to know about Google Glass*, en. [Online]. Available: https://www.washingtonpost.com/news/the-switch/wp/2014/02/27/everything-you-need-to-know-about-google-glass/.

[10] Microsoft, *Microsoft announces global expansion for HoloLens*, en-AU, Oct. 2016. [Online]. Available: https://news.microsoft.com/en-au/2016/10/12/microsoft-announces-global-expansion-for-hololens/.

[11] *Wearable Technology in Medicine and Health Care*. [Online]. Available: https://books.google.com/books/about/Wearable_Technology_in_Medicine_and_Heal.html?id=9aRBDwAAQBAJ.

[12] C. Wang, W. Lu, M. R. Narayanan, S. J. Redmond, and N. H. Lovell, "Low-power technologies for wearable telecare and telehealth systems: A review," en,

Biomedical Engineering Letters, vol. 5, no. 1, pp. 1–9, Mar. 2015, ISSN: 2093-9868, 2093-985X. DOI: 10.1007/s13534-015-0174-2. [Online]. Available: http://link.springer.com/10.1007/s13534-015-0174-2.

[13] M. P. Turakhia, M. Desai, H. Hedlin, A. Rajmane, N. Talati, T. Ferris, S. Desai, D. Nag, M. Patel, P. Kowey, J. S. Rumsfeld, A. M. Russo, M. T. Hills, C. B. Granger, K. W. Mahaffey, and M. V. Perez, "Rationale and design of a large-scale, app-based study to identify cardiac arrhythmias using a smartwatch: The Apple Heart Study," en, *American Heart Journal*, vol. 207, pp. 66–75, Jan. 2019, ISSN: 00028703. DOI: 10.1016/j.ahj.2018.09.002. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0002870318302710.

[14] J. Mischke, *Wearable Technology: The Latest Trend in Professional Sports*, en-US, Text, May 2018. [Online]. Available: https://www.wearable-technologies.com/2018/05/wearable-technology-the-latest-trend-in-professional-sports/.

[15] G. Udovicic, A. Topic, and M. Russo, "Wearable technologies for smart environments: A review with emphasis on BCI," in *2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Croatia: IEEE, Sep. 2016, pp. 1–9. DOI: 10.1109/SOFTCOM.2016.7772186. [Online]. Available: http://ieeexplore.ieee.org/document/7772186/.

[16] K. E. Friedl, "Military applications of soldier physiological monitoring," en, *Journal of Science and Medicine in Sport*, vol. 21, no. 11, pp. 1147–1153, Nov. 2018, ISSN: 14402440. DOI: 10.1016/j.jsams.2018.06.004. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S144024401830255X.

[17] Deloitte, *Future of mining with wearables*, en. [Online]. Available: https://www2.deloitte.com/ca/en/pages/energy-and-resources/articles/future-mining-wearables-improve-safety.html.

[18] Mokhinabonu Mardonova and Yosoon Choi, "Review of Wearable Device Technology and Its Applications to the Mining Industry," en, *Energies*, vol. 11, no. 3, p. 547, Mar. 2018, ISSN: 1996-1073. DOI: 10.3390/en11030547. [Online]. Available: http://www.mdpi.com/1996-1073/11/3/547.

[19] CSIRO, *Fitness tracker for cows to help Aussie farmers*, en-AU. [Online]. Available: https://www.csiro.au/en/News/News-releases/2018/Fitness-tracker-for-cows-to-help-Aussie-farmers.

[20] R. Ali and A.-S. Pathan, "The state-of-the-art wireless body area sensor networks: A survey," *International Journal of Distributed Sensor Networks*, vol. 14, p. 155 014 771 876 899, Apr. 2018. DOI: 10.1177/1550147718768994.

[21] S. Lee, B. Grundlehner, R. Garcia van der Westen, S. Polito, and C. Van Hoof, "Nightingale v2: Low-power compact-sized multi-sensor platform for wearable health monitoring," Jul. 2019, pp. 1290–1293. DOI: 10.1109/EMBC.2019.8856847.

[22] M. Ballerini, M. Magno, D. Brunelli, G. Cornai, and L. Benini, "Netwis: A scalable and robust body sensor network for biomedical application," Jun. 2019, pp. 118–123. DOI: 10.1109/IWASI.2019.8791326.

[23] D. Roggen, A. Pouryazdan, and M. Ciliberto, "Poster: Bluesense - designing an extensible platform for wearable motion sensing, sensor research and iot applications," in *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*, Madrid, Spain: Junction Publishing, 2018, pp. 177–178, ISBN: 978-0-9949886-2-1. [Online]. Available: http://dl.acm.org/citation.cfm?id=3234847.3234874.

[24] M. Ghamari, "A review on wearable photoplethysmography sensors and their potential future applications in health care," en, *International Journal of Biosensors & Bioelectronics*, vol. 4, no. 4, 2018, ISSN: 25732838. DOI: 10.15406/ijbsbe.2018.04.00125. [Online]. Available: https://medcraveonline.com/IJBSBE/IJBSBE-04-00125.php (visited on 10/11/2019).

[25] Polar, *Polar H10 — Heart rate monitor chest strap — Polar Canada*. [Online]. Available: https://www.polar.com/ca-en/products/accessories/h10_heart_rate_sensor.

[26] J. Friel, *Total Heart Rate Training: Customize and Maximize Your Workout Using a Heart Rate Monitor*, en. Ulysses Press, Jun. 2009, Google-Books-ID: 6pc2phOsfdwC, ISBN: 978-1-56975-389-7.

[27] Empatica, *Real-time physiological signals — E4 EDA/GSR sensor*, en-us. [Online]. Available: https://www.empatica.com/research/e4.

[28] BioRadio, *BioRadio Specifications —*. [Online]. Available: https://glneurotech.com/bioradio/bioradio-specifications/.

[29] ShimmerSensing, *Consensys Bundle Development kit — Complete Wearable Sensor Kit — IMU - ECG - EMG - GSR*. [Online]. Available: https://www.shimmersensing.com/products/consensys-ecg-development-kits-update.

[30] Murata, *MBN52832*, en. [Online]. Available: https://wireless.murata.com/mbn52832.html.

[31] N. Semiconductor, *nRF52832 - Nordic Semiconductor*, en. [Online]. Available: https://www.nordicsemi.com/en/Products/Low%20power%20short-range%20wireless/nRF52832.

[32] ——, *Nrf52832 - product specification v1.0*, en. [Online]. Available: https://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.0.pdf.

[33] D. Incorporated, *Dual 150ma low quiescent current fast transient low dropout linear regulator*, en. [Online]. Available: https://www.diodes.com/assets/Datasheets/AP7312.pdf.

[34] M. Integrated, *On/off controller with debounce and +/- 15kv esd protection*, en. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX16054.pdf.

[35] STMicroelectronics, *Inemo inertial module: Always-on 3d accelerometer and 3d gyroscope*, en. [Online]. Available: https://www.st.com/resource/en/datasheet/lsm6ds3.pdf.

[36] ——, *STMicroelectronics*, en. [Online]. Available: https://www.st.com/content/st_com/en.html.

[37] T. Instruments, *Drv8835 dual low-voltage h-bridge ic*, en. [Online]. Available: http://www.ti.com/lit/ds/symlink/drv8835.pdf.

[38] SnapEDA, *SnapEDA*, en. [Online]. Available: https://www.snapeda.com.

[39] SEGGER, *Interface Description*, en. [Online]. Available: https://www.segger.com/products/debug-probes/j-link/technology/interface-description/.

[40] Tag-Connect, *Tag Connect*. [Online]. Available: http://www.tag-connect.com/.

[41] Adafruit, *Bluefruit nRF52 Feather Learning Guide*, en-US. [Online]. Available: https://learn.adafruit.com/bluefruit-nrf52-feather-learning-guide/downloads.

[42] Raytac, *Mdbt42*, en. [Online]. Available: https://www.raytac.com/download/index.php?index_id=26.

[43] S. Labs, *Single-chip usb-to-uart bridge*, en. [Online]. Available: https://www.silabs.com/documents/public/data-sheets/cp2104.pdf.

[44] T. Instruments, *Bq2416xx 2.5a, dual-input, single-cell switched-mode li-ion battery charger with power path management and i2c interface*, en. [Online]. Available: http://www.ti.com/lit/ds/symlink/bq24160.pdf.

[45] PCBCart, *PCB Manufacturing and Assembly All in One Place — PCBCart*. [Online]. Available: https://www.pcbcart.com/.

[46] Digikey, *DigiKey Electronics Canada*. [Online]. Available: https://www.digikey.ca/.

[47] Arduino, *Arduino/Arduino*, Oct. 2019. [Online]. Available: https://github.com/arduino/Arduino.

[48] Nordic Semiconductor, *S132 SoftDevice*, en. [Online]. Available: https://www.nordicsemi.com/en/Software%20and%20Tools/Software/S132.

[49] FreeRTOS, *The freertos kernel market leading, de-facto standard and cross platform rtos kernel.* en-US. [Online]. Available: https://www.freertos.org/.

[50] Adafruit, *Bluefruit52*, en. [Online]. Available: https://github.com/adafruit/Adafruit_nRF52_Arduino.

[51] The Qt Company, *Qt — Cross-platform software development for embedded & desktop*, en. [Online]. Available: https://www.qt.io.

[52] K. Science, *Kst - Visualize your data*. [Online]. Available: https://kst-plot.kde.org/.

[53] O. Project, *Open Broadcaster Software — OBS*. [Online]. Available: https://obsproject.com/.

[54] bitHeads, *bitHeads announces brainCloud*, en-CA, Nov. 2014. [Online]. Available: https://getbraincloud.com/apidocs/bitheads-announces-braincloud/.

[55] Autodesk, *Cloud Powered 3d CAD/CAM Software for Product Design Fusion 360*, en-CA. [Online]. Available: https://www.autodesk.ca/en/products/fusion-360/overview.

[56] J. Prusa, *Original Prusa i3 MK3*, en-US. [Online]. Available: https://www.prusa3d.com/original-prusa-i3-mk3/.

[57] Y. Liang, C.-Z. Zhao, H. Yuan, Y. Chen, W. Zhang, J.-Q. Huang, D. Yu, Y. Liu, M.-M. Titirici, Y.-L. Chueh, H. Yu, and Q. Zhang, "A review of rechargeable batteries for portable electronic devices," en, vol. 1, no. 1, pp. 6–32, 2019, ISSN: 2567-3165. DOI: 10.1002/inf2.12000. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/inf2.12000.

[58] M. Boric, A. F. Vilas, and R. P. D. Redondo, "BLE broadcasting impact in a real network environment," en, in *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing - ICC '17*, Cambridge, United Kingdom: ACM Press, 2017, pp. 1–6, ISBN: 978-1-4503-4774-7. DOI: 10.1145/3018896.3018949. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3018896.3018949.

[59] M. B. Aimonetto and F. Fiori, "Susceptibility of 2.4ghz low-power receivers to low frequency EMI," in *2018 IEEE International Symposium on Electromagnetic Compatibility and 2018 IEEE Asia-Pacific Symposium on Electromagnetic Compatibility (EMC/APEMC)*, May 2018, pp. 782–787. DOI: 10.1109/ISEMC.2018.8393888.

[60] BeamNG, *BeamNG.drive*, en-US. [Online]. Available: https://www.beamng.com/.

[61] MATLAB, *MATLAB - MathWorks*, en. [Online]. Available: https://www.mathworks.com/products/matlab.html.

[62] I. Bluetooth SIG, *Bluetooth specifications*, en. [Online]. Available: https://www.bluetooth.com/specifications/.

[63] M. C. Eugenia Cabot, *The effect of the human body on wireless microphone transmission*, en. [Online]. Available: https://d24z4d3zypmncx.cloudfront.net/KnowledgeBaseFiles/cjrhnpgx70set01892ltnfjmr/Effect_Human_Body_Wireless_Mic_Transmission.pdf.