Hand Gesture Recognition Through Capacitive Sensing

A thesis presented in partial fulfilment of the
requirements for the degree of

Master of Engineering

in

Electronics & Computer Engineering

at

Massey University,

School of Food and Advanced Technology (SF&AT)

Auckland, New Zealand

Muqing Xu
February 2022

ACKNOWLEDGEMENTS

I would like to acknowledge and give my greatest gratitude to my supervisor Associate Professor Fakhrul Alam and co-supervisor Dr Frazer Noble, who made this project a possibility with their constant support and guidance.

Secondly, thank you to Kay Lal for his generous assistance and support in providing technical assistance throughout the project.

Thank you to Massey University in providing the funding and financial assistance whenever required, without which this research would not have been possible.

Finally, I would like to thank the technical and administrative staff at Massey University for creating such an outstanding research environment that supported me throughout my studies.

ABSTRACT

This thesis investigated capacitive sensing-based hand gesture recognition by developing and validating through custom built hardware. We attempted to discover if massed arrays of capacitance sensors can produce a robust system capable of simple hand gesture detection and recognition.

The first stage of this research was to build the hardware that performed capacitance sensing. This hardware needs to be sensitive enough to capture minor variations in capacitance values, while also reducing stray capacitance to their minimum. The hardware designed in this stage formed the basis of all the data captured and utilised for subsequent training and testing of machine learning based classifiers.

The second stage of this system used mass arrays of capacitance sensor pads to capture frames of hand gestures in the form of low-resolution 2D images. The raw data was then processed to account for random variations and noise present naturally in the surrounding environment. Five different gestures were captured from several test participants and used to train, validate and test the classifiers.

Different methods were explored in the recognition and classification stage: initially, simple probabilistic classifiers were used; afterwards, neural networks were used. Two types of neural networks are explored, namely Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN), which are capable of achieving upwards of 92.34 % classification accuracy.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Problem and Research Aim

Hand gestures can be argued as a fundamental method of how humans interact with their surroundings. Hand gestures can be categorized into multiple different classes; controlling gestures, conversation gestures, communicative gestures, and many others. Among various hand gesture recognition methods for Human-Computer Interaction (HCI), camera-based ones are the most common. The motivation behind such research includes applications such as sign-language recognition, robotic control, directional indication, immersive game technology, virtual controllers, affective computing, and remote controlling [1]. In all these applications, gesture recognition is a critical component and directly impacts on user-experience. However, despite the above mentioned potential, hand gestures recognition has not been as thoroughly researched as other types of HCI, such as voice recognition.

Recent research in hand gestures has improved classification results through the use of novel methods of signal processing, detection, tracking, shape description, motion analysis, and pattern recognition. While there are enormous strides in software development, the fundamental way we attempt to process gestures through computational means has remained the same: computer learning through information captured from a piece of physical hardware or sensor. Currently, most hand gesture data and training data sets are captured through the use of video cameras since the continued advancement in camera technology allows us to capture larger and larger resolution images. While the usage of the camera has proven successful, using a video camera to capture hand gesture data is sometimes "wasteful". Cameras capture images (data frames) in high-resolution RGB images, which also contains background imagery features that are unnecessary for gesture recognition. This background noise is ever-present and must be actively filtered to distinguish noise from features before gesture recognition can be performed. This large amount of incoming data affects the speed of processing, especially for high-resolution images.

Many simple gesture recognition applications do not require such extensive resolution to perform and only require basic features to differentiate between gestures; high-resolution RGB imagery is not needed in such cases. In fact, for simple gesture recognition, such as static recognition, much of the data captured from video cameras is processed out as not to overtrain models. In such cases, many RGB images are routinely turned into greyscale images before being classified.

This thesis explores an alternative method of hand gesture recognition by using capacitive sensors. Similar to camera-based sensing, capacitance sensing is contactless and therefore, hygienic and offers freedom of hand movement. Capacitive sensors naturally produce 2D "greyscale", low-resolution images and require very little power to capture data. It has low processing requirements and is ideal when processing power and power consumption are critical. The capacitive sensors also work in low-light conditions, where a camera system would struggle to capture useful data. This offers much higher privacy than camera-based systems, as capacitor sensors are physically incapable of capturing personal information such as faces and fingerprints.

While using capacitive sensors for gesture recognition is not an entirely new concept, most reported methods are restricted to straightforward designs with few capacitance sensors and operate in threshold detection mode. In such cases, capacitance sensors are used as on/off threshold detection to deduce if a physical body part is detected. Essentially, classification is performed with monotone images. This thesis attempts to advance this by using more capacitor sensors and not operating in threshold detection mode. The approach proposed is to create a low-resolution grayscale "image" composed of capacitance values and then use the images to train and test classifiers to conclude the viability of capacitive sensing as a form of free air gesture recognition.

1.2 Recent Progress in Gesture Recognition Technology

As this work is based on capacitance-based gesture recognition, we will first look at current hand gesture recognition technology and how capacitance sensing works.

The first step for any data collection/analysis system is to collect the raw data necessary to accomplish the analysis. A range of different approaches has been employed in this raw data collection stage for hand gesture recognition. The most common of those being image-based approaches, such as depth sensors, single cameras, and non-image-based systems, such as gloves and bands [2; 3]. Different groups have also explored other more novel non-image technologies techniques such as RF signal-based sensors by Google [4] or capacitive sensing and ultrasound-based systems. All of these new techniques are usually classified as non-image-based and non-wearable. From the different types of technology, we can roughly classify gesture recognition technology into image-based systems and non-image-based systems, as shown in Figure 1.1.

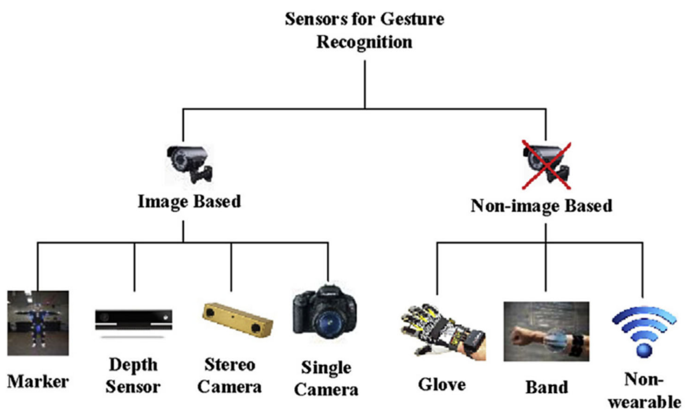


Figure 1.1: *Different types of gesture recognition sensors [3]*

1.2.1 Image-based Systems:

An image-based system is a natural extension of a human's ability to decipher gestures. It is assumed that the human eye can be replaced with cameras that can “see” a gesture [3]. Although this sounds simple in practice, there are many challenges to overcome by such an approach. Background scenery adds unwanted noise, lighting variation causes variations between samples, and hand size variations all contribute to this difficulty. Locating the hand and segmenting it from a busy background in an image sequence is non-trivial [5], adds computational cost and reduces the robustness of the system [6]. Additional support, such as markers, depth sensors, and other supporting instruments, are used so that the computer can better separate noise and increase the system's robustness [2].

1.2.1.1 Single camera

A single camera represents the simplest form of an image-based system, where the hardware consists of only a single camera. Single camera-based gesture recognition systems first appeared in the early ’90s [7]. While they were simple and cost-effective, they had difficulty classifying gestures with “busy” backgrounds. Even today, single cameras still capture 2D images by nature, therefore separating background noise from gesture is computationally expensive. This effect is compounded when the gesture is non-static, and real-time recognition becomes challenging as vast amounts of data must be processed frame by frame [8].

1.2.1.2 Markers

Makers represent the next logical approach in addressing the issue of the difficulty of separating gesture data from background noise. In such approaches, users need to wear obvious markers to identify features from the background. Since these markers are easily identifiable by the computer, it increases the system's robustness by “telling” a computer where to look for the gesture data [9]. The downside to this is that the user must wear obvious markers, or the system will cease to function. In recent times, with the rise of faster graphical processing speed and more advanced recognition systems, the use of markers has declined as it does not provide a significant enough gain in classification accuracy compared to non-marker approaches [3] to justify the added movement burden of the additional hardware.

1.2.1.3 Stereo Camera

A stereo camera has the ability to mimic human binocular vision and is therefore, able to capture three-dimensional (3D) images. Stereoscopic camera systems capture two simultaneous images from a pair of calibrated video cameras and use image registration methods to create a disparity map that approximates per-pixel depth [5]. Due to the 3D nature of the captured images, we can increase the robustness as background noise can be more easily separated from the gestures. This, however, comes at the cost of increased hardware expense and higher computational overhead to solve the image registration for each image pair [5].

Many research works have explored 3D cameras with varying success [10]. Other forms of 3D cameras also exist that offer even higher accuracy and image fidelity, such as Time of Flight (ToF) cameras. TOF cameras directly provide the “depth” information in the form of a data stream whereas, for the stereo camera, the end-user must calculate the information themselves [10; 11]. However, normal stereo cameras work well in bright light and can be built with standard video cameras, whereas ToF requires specialized equipment and does not work well in bright light conditions [3; 11].

1.2.1.4 Depth Sensor

The depth sensor is a non-stereo, image-based depth sensing device. The depth sensor provides a 3D depth output. Depth sensors enjoy several advantages compared to the traditional stereo camera approach [3]. For example, the problems of setup, calibration, and illumination conditions can be prevented [5]. Multiple implementations of camera systems with depth sensors can be found, such as Microsoft’s Kinect, which features a QVGA (320x240) depth camera and a VGA (640x480) video camera, both of which produce image streams at 30 Frames Per Second (FPS) [5].

1.2.2 Non-Image-based Systems

In the past, image-based systems have primarily dominated the gesture recognition discipline due to their robustness [3]. While non-image-based systems, such as gloves and band-based systems, are available, their practical applications have remained limited [3]. It is only in recent times that alternative methods of non-intrusive hardware design, such as capacitive sensor, ultrasound sensor, and RF sensor, are being explored. This growth in non-image-based gesture recognition allows hardware designs specifically targeted at gesture recognition.

1.2.2.1 Gloves

Glove design involves the end-user wearing a “glove” with sensors attached at critical locations, which can record their hand configuration/motion [12], as shown in Figure 1.2 these sensors are usually in the form of accelerometers and gyroscopes, which record user motion and speed. These devices also typically require a physical connection to the computer to enable data transfer [3]. While the glove-based system is not favourable in applications where the end-user is concerned due to the restricted nature of the glove system, it is a very popular choice for hand movement acquisition due to the precisions such systems can achieve [12]. A range of different gloves was proposed in the past. Few such devices enjoyed commercial success, showing that glove design has its niche use cases. Below are three examples of successful glove designs:

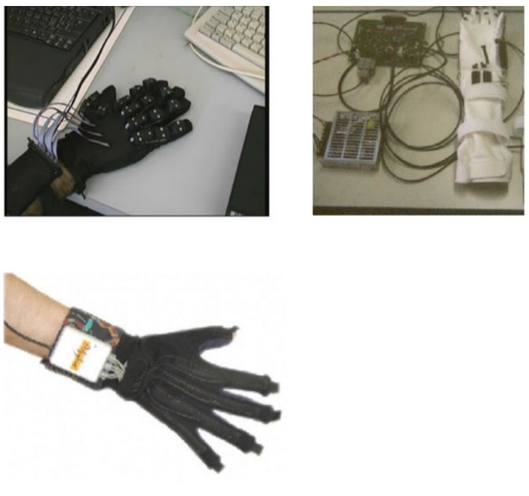


Figure 1.2: *Examples of Glove based designs; top left: Humanglove. Top right: strinGlove. Bottom left: Didjiglove*

1) Humanglove: Equipped with 20 Hall effect sensors that measure flexion/extension of the four fingers, Metacarpophalangeal joint (MCP), Proximal interphalangeal joint (PIP) and Distal interphalangeal joint (DIP) joints and flexion/extension of the thumb, thumb metacarpophalangeal joint (TMCP) and metacarpal phalangeal (MP), and

Interphalangeal (IP) joints, as well as fingers and thumb abduction/adduction; two additional sensors measure wrist flexion and abduction/adduction [13].

2) StrinGlove: Uses 24 inductcoders (inductance based coders) [14] to obtain 22 degrees of freedom of human hand to record MCP, PIP, DIP angles of fingers and MP and IP angles of thumb, abduction/adduction angles of fingers and thumb, as well as wrist motion [15]. It is also equipped with nine contacts (magnetic) sensors, placed one in the thumb and two on each finger (tip and PIP phalanx). It requires calibration [12; 16].

3) Didjiglove: uses ten capacitive bend sensors to record finger flexion (fingers MCP and PIP, and thumb TCMP and MP). The sensor employs two layers of conductive material separated by a dielectric. When a finger bends, it would change the overlapping conductive layers and thus a change in capacitance value. Calibration before each use is required for proper function. The device was initially designed as a form of capturing human animation [12; 15; 16].

1.2.2.2 Band

Band sensing technology is similar to glove technology in that it requires the user to wear a device. However, unlike gloves, the device is not as "intrusive". The device allows total movement of the hand without restriction. Figure 1.3 shows one such device called Tomo, which was developed by Zhang, Y. & Harrison, C. [17].

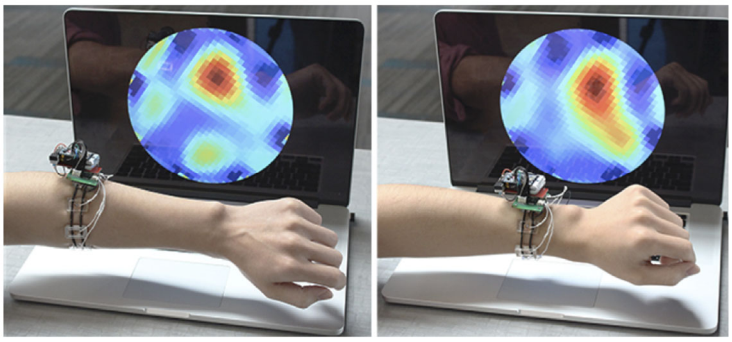


Figure 1.3: *Tomo, tomography-based hand gesture recognition hardware [17]*

Tomo is based on using electrical impedance tomography, which is able to capture the impedance geometry of a user's arm. This impedance geometry mapping is achieved by measuring the cross-sectional impedances between all pairs of eight electrodes resting on a user's skin [17]. This device assumes that the impedance map between each gesture is different and therefore classifiable when performing different gestures. This approach is one of the more novel ideas. Other improvements on this idea have attached a camera to the band as a form of secondary data capture, which allows for more precise and rapid gesture classifications [18].

CHAPTER 2: CAPACITIVE SENSING

2.1 Introduction

This chapter explores why we chose to use capacitive sensing as a form of gesture recognition and the advantages and disadvantages of using captive sensing. The background of capacitive sensing and its operation principles will also be discussed.

2.2 Capacitance Sensing Background

The use of capacitive sensing is not new and is extremely widespread. Touchscreens, touchpads, and capacitive "buttons" are all frequently used with capacitive sensing technology [19; 20]. Most of these devices rely on physical contact with a surface to register a "touch". Attempts to use capacitive sensing on more sophisticated non-contact based sensing human-computer interactions have also been explored, such as body parameter sensing and gesture recognition but to a lesser degree compared to surface touch solutions. At the same time, attempts to replace input modalities, such as the keyboard and mouse, with capacitive sensing are also explored. However, most of such designs are still in a developing stage [1].

While the use of touch-based capacitance sensing has been heavily explored, the research into close-range, non-contact capacitance-based sensing remains fragmented by different approaches, both in design and implementation. This effect is further compounded by the fact that most non-contact sensing designs require extensive software and machine learning, which further complicates the matter [19].

2.2.1 Capacitance Sensing Advantages

The main advantages that capacitive sensing has over other detection approaches are that it can sense different kinds of materials (skin, plastic, metal, liquid), is contactless, and is wear-free [21]. Privacy of capacitive systems is also naturally higher compared to camera based systems. Capacitance sensing hardware is simple, cheap by nature, can be easily manufactured, and often occupies less space than other forms of wireless sensing hardware [19]. They typically have a minimum height profile and can be hidden under opaque, non-conductive materials, and can be arranged in large, high-resolution scanned arrays. Capacitive sensing is purely electrical and requires low power. No form of mechanical movement is required for capacitive sensing [19].

2.2.2 Capacitance Sensing Disadvantages

The biggest disadvantages that capacitive sensing has to include sensing distance, measurement drifts and interference. Compared to other sensors capacitive sensors drop in accuracy much more quickly [20] and is very much sensitive to environmental changes such as temperature, humidity, etc., which will affect the final performance and accuracy of the sensors. Capacitance also is very sensitive to interference and noise due to the nature of the small capacitance (femto farads) values involved. All of the above can be mitigated to a certain degree through software means and the absolute accuracy of capacitance sensors is not crucial in the use case of gesture recognition.

2.3 Capacitive Sensing Principles

Capacitive sensing can be achieved due to inherent capacitances between the people, devices, and objects in the physical environment around us [19]. As shown in Figure 2.1, a range of natural capacitance exists between different objects; in particular, there is a capacitance between the human body and all surrounding objects. These capacitances are not physical capacitors but are instead caused by capacitive coupling between different objects. By measuring or detecting these values, we are able to calculate and relay information, such as positions and motions [22].

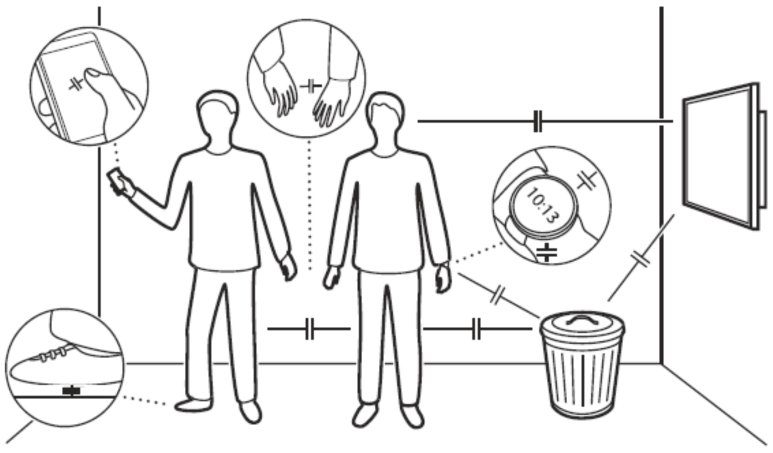


Figure 2.1: *Capacitance that exists naturally between different objects [19]*

It should be noted that the terms "capacitive sensing" and "electric field sensing", in most cases, refer to the same basic technique [19]. Throughout this thesis, we use the former term.

Figure 2.2 shows a lumped circuit model of what a capacitive sensing circuit looks like and is used to describe what is actually being measured by capacitance sensors. As shown, two electrodes T and R are used in the circuit; from these two electrodes, different capacitances will exist between *T*, *R,* and the hand *H*. Depending on the sensing mode of the circuits, one or more of $C_0, C_1, and\ C_2$, will be measured to produce and relay information relating to the location of the hand [19]. The materials used for electrodes T and R vary widely, but are typically conductive materials, such as copper, novel non-solid parts such as textiles foils, paints, and printable conductive materials have also been used [23].

Figure 2.2: *Lumped circuit model of capacitive sensing [24]*

2.3.1 Capacitance Sensing Operation Modes

One of the first examples of gesture recognition through capacitive sensing is the “Theremin” [25]. The device, a musical instrument, invented by Russian physicist Leon Theremin in 1920, was able to play music without physical contact with the instrument. The instrument had two antennas that could detect the relative proximity of hands. The player can vary the pitch and volume of the instrument by varying the distance of each hand to the antennas [26]. This device showed that as early as 1920, touchless sensing was being actively explored.

In 1995 Zimmerman et al. proposed different taxonomy to describe different types of operating modes used in capacitive sensing [27]. They first proposed the idea of active capacitive sensing: the human Shunt mode [27]. This work was later extended to introduce the concept of passive capacitor sensing [28]. While dynamic sensing systems must actively generate an electric field, passive sensing systems rely on external or ambient electric fields that are passively sensed. The advantage of passive sensing is that it is simpler and requires less space per sensor pad. The disadvantage of passive sensing is that it is less accurate and affected by environmental changes more than active modes [19; 28].

In 1999, Smith consolidated both active and passive approaches and classified capacitance sensing into three distinct modes: loading mode, transmit mode and shunt mode *[20; 29]*. Of which, the loading mode is passive, and other modes are active.

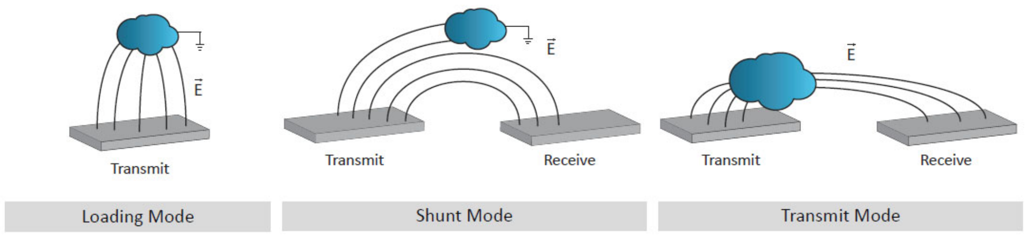


Figure 2.3: *modes of operation proposed by J. Smith [20; 29]*

2.3.1.1 Loading Mode

Loading mode, is, arguably, the simplest mode as it only requires a single plate that acts both as transmit and receiver [20; 29]. Loading mode is passive by nature as it relies on measuring the displacement current caused by the presence of a grounded object in proximity of the single electrode. Referring to Figure 2.2, loading mode would mean only the electrode T would exist. Therefore, the only significantly useful capacitance would be C_1, which exists between the hand and the electrode [29]. As loading mode has only a single electrode, for n number of capacitive sensors (electrodes), only n amount of measurements/data points can be made. The advantage of this mode is that it only needs a single electrode, can be placed arbitrarily and be more easily shielded from influences of existing electric potentials than other modes [20; 30]. It also has the ability to detect electric field changes in longer ranges than the other modes, although with less accuracy and repeatability [29; 31].

2.3.1.2 Shunt Mode

In shunt mode, shown in Figure 2.3, a known voltage of a certain frequency is created between the two electrodes to cause displacement current to flow. This requirement of a displacement current means that one electrode will be required to transmit a predetermined signal in shunt mode [27]. When a hand is in close proximity, it will capacitively couple to both the transmitting and receiving electrodes and cause some of the displacement current to flow from the transmitting electrode through the hand to the ground. By measuring the decrease in displacement current that is being received by the receive electrode, one can determine the hand's distance in relation to the capacitive sensor.

Shunt mode offers a higher spatial resolution than the loading method but does not necessarily provide larger detection distances [20]. Shunt mode will also offer more possible measurement points with the same number of sensors as loading mode. Whereas loading mode can only produce n amount of measurement points with n amount of sensors, shunt mode is able to produce $\frac{n*(n-1)}{2}$ amount of measurement points [32]. However, unlike loading mode, where each of the electrodes can be placed arbitrarily, the positioning of electrodes in shunt mode plays a critical role in measurement accuracy and repeatability [33]. Different layouts of shunt mode have been explored in various works e.g., grid styles, hexagons SnakePit, and CellMatrix [34; 35]. This grid style design has also been extensively exploited to yield high-resolution sensing that is being widely used in today's commercial touchscreens [19; 35; 36].

2.3.1.3 Transmit Mode

Transmit mode can be considered to be a special form of Shunt mode. In Transmit mode, the hand is treated as a tightly coupled object to the transmitter plate. This means that the coupling between the hand and the transmitter is much greater than the coupling between the hand and the receiver or between the transmitter and the receiver [19]. Therefore, the receiver plate would see more displacement current [29]. Referring back to Figure 2.2, as the hand approaches the receive electrode *R*, the value of C_2 (and C_0 - the two are not distinct in this mode) increases, and the displacement current received at R increases [29; 32]. The patterns used for transmit mode also tend to closely match the styles found on shunt modes [37].

While shunt mode offers a higher spatial resolution than loading mode, it comes at the price of increased surface area per sensor[20]. When a single sensor is employed to measure water levels, accuracy is paramount to offer accurate values and sensor size is not crucial. However, in the case of gesture sensing, we must take into consideration the pad size and the pad count. The number of sensors is paramount for sensing applications as it affects the resolution of the "picture" generated for use with machine learning. Therefore, for this research, loading mode was used as it offered the benefit of the smallest required pad space, whereas shunt mode required two separate sensing pads, which increased the sensing pad surface area.

2.4 Capacitance Sensing Operation Distance

The goal of any capacitance sensor is to detect an object of interest, in our case the human hand, at the furthest distance and with the highest spatial resolution. Spatial resolution here means the size of the smallest object that the capacitance sensors can resolve.

While it is known that the human body produces capacitance, the actual amount of capacitance can vary widely from person-to-person with different body mass, shape, and different measurement conditions [22]. The issue is compounded by the fact that individual capacitance sensors also produce varying results depending on the material and size of the sensor [20]. Literature suggests that a real-world detectable human capacitance is on the order of low hundreds of picofarads [19; 22; 38]. At this level of capacitance, noise starts to be a dominating factor; if the design does not account for stray capacitance and component selection/routing becomes an essential factor. Thankfully many commercially available Integrated Circuits (ICs) exist to measure capacitances at such small level. For example, the FDC1004 [54] is a simple, cost-effective capacitance-to-digital converter that can achieve a resolution of 0.5 fF and was chosen to construct the capacitance sensors for this project.

One of the critical challenges of capacitance sensing is the quick decaying detectable capacitance rate in relation to the detection range. The generic equation for the capacitance of a capacitor is given as $C \approx \frac{\epsilon A}{D}$, where A is the pad area, D is the distance from the pad, and C is the measured capacitance. Therefore, this would suggest that capacitance is inversely related to distance. However, this is not the case for greater distances between the plates. The farther apart the two plates are, the smaller their overlapping area gets relatively to their surroundings; thus, a more realistic model is $C \approx \frac{\epsilon A}{D^x}$, where x is dependent on the environment [39]. From this equation, it can be concluded that since capacitance decays exponentially, the detectable range of capacitor sensors is closely related to the size of the sensor [20; 40].

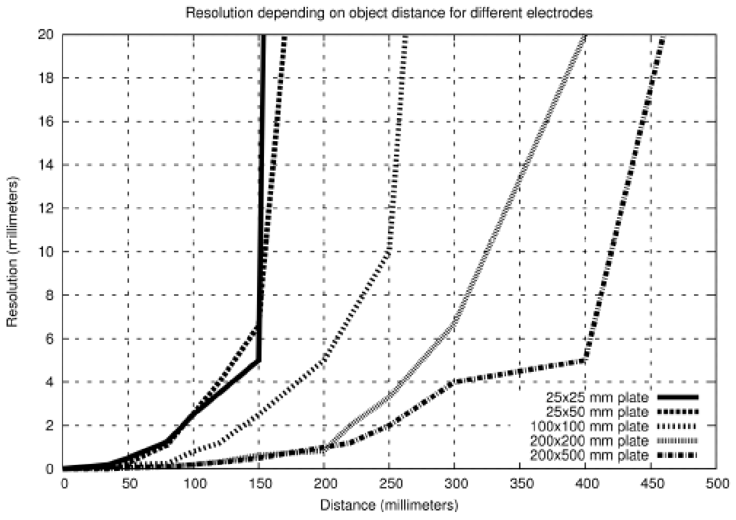


Figure 2.4: *Comparison of CapBoard sensitivity for different sensor sizes [39]*

Figure 2.4 illustrates the exponential decay of different sized sensors which all have a relatively linear region and then rapidly drop in resolution verse distance [39].

For our design, we have opted to use a rectangle pad design with a pad size of 27.2 mm by 5.1 mm. With this pad size we are able to have a total sensing area of 137.7 mm^2, and a spatial resolution of approximately 55.0 mm. At this range, our design is still capable of differentiating between two fingers. From above 55.0 mm, the detection rapidly becomes exponential decay and becomes undifferentiable to environmental noise.

CHAPTER 3: DESIGN AND PROPOSED SYSTEM

3.1 Introduction

This chapter explores the hardware choices for our design and how we came to use such designs. We first look at different sensor pad sizes that would affect the spatial resolution of our data capture. Based upon the above results we will then propose and design our hardware system. Component choices will be backed-up with our design philosophy and reasonings.

3.2 Proposed System Overview

Physical pictures of the proposed system are shown in Figure 3.1. The white board contains the sensor pads that are connected to the Motherboard (the green PCB). When a hand approaches the sensor pads, loading mode capacitance sensing is achieved which the FDC1004 will actively measure.

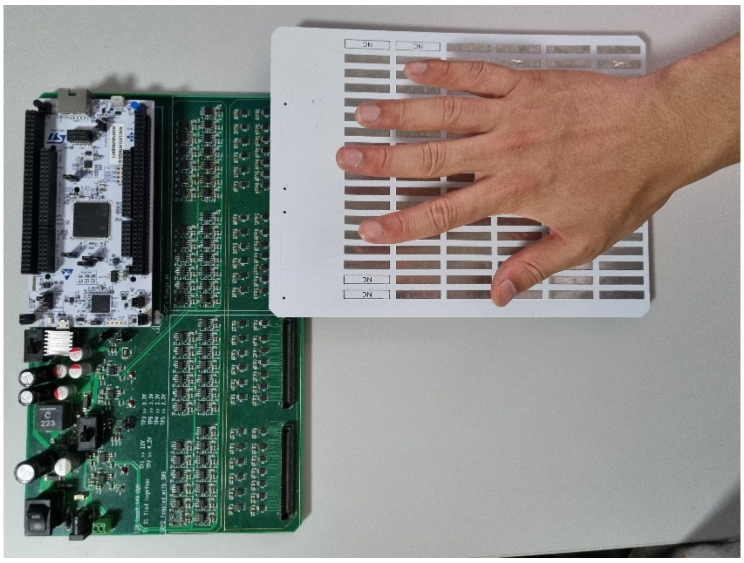


Figure 3.1: *Complete mated hardware solution*

The high-level flow diagram of the system is shown in Figure 3.2. The Motherboard handles the data collection, processing and delivery to the host PC, while the sensor pad board is the physical medium on which data is collected.

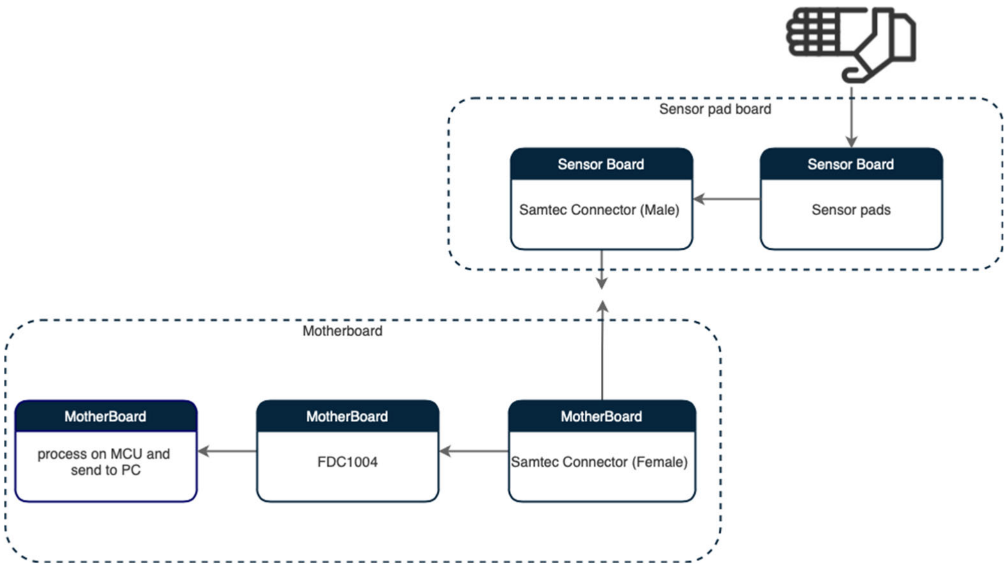


Figure 3.2: *High-level flow diagram of the proposed system*

The sensor pad board was designed to be replaceable to allow variations in pad size. By separating the sensor pad board and the Motherboard, we made the sensor pad board replaceable which allows us to test and validate different sensor pad sizes/arrangements. Data is collected by the Motherboard and sent to the host PC through a USB serial. The detached sensor pad board and the Motherboard are shown in Figure 3.3.

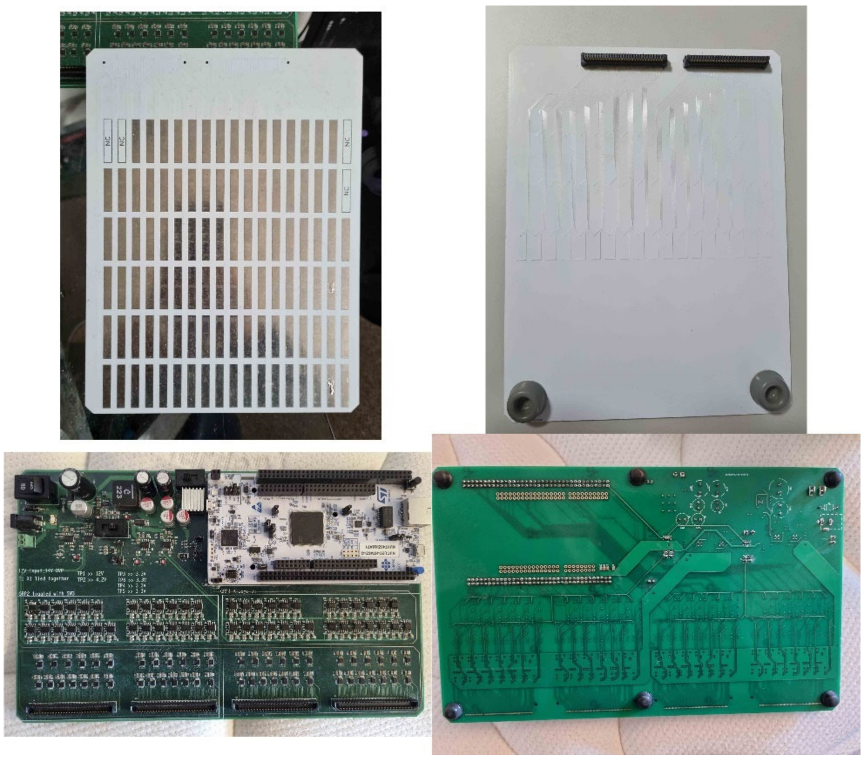


Figure 3.3: *Top left: top-side of the sensor board, Top-Right: bottom-side of the sensor board, Bottom Left: top-side of Motherboard, Bottom Right, bottom-side of Motherboard*

3.3 Motherboard Design and Component Choice

The motherboard is a four-layer PCB design, with FR-4 as the PCB material. The design was restricted to four layers due to costs reasons. For ease of design and documentation, the prototyping board can be split into four major sections: 1) power; 2) STM32F7; 3) sensor section; and 4) connectors.

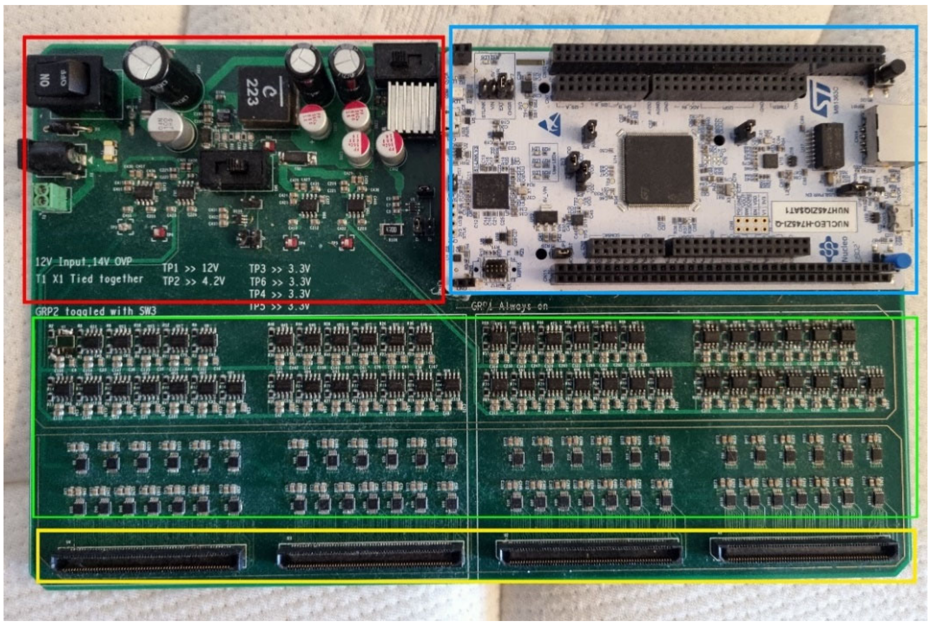


Figure 3.4: *Major Board sections*

From Figure 3.4, we can see all the major motherboard sections. Red represents the power stage, which is responsible for all the power supply rails of the entirety of the Motherboard; blue is the STM32F7 MCU, which is used for processing, calculations and communication with the host PC; green is the FDC1004 capacitance to digital converter and STM32G03 MCUs used for collecting capacitance data from the sensor pads and transferring said data to the STM32F7; and yellow outlines the female Samtec connectors, which are used to connect to the sensor board (explained in section 3.4.3).

3.3.1 MCU and Capacitance Sensor

A challenge in designing the Motherboard was to fit sufficient capacitance sensors (FDC1004) to enable a high enough resolution design to support our use case. Off-the-shelf capacitance sensor FDC1004 [21] by Texas Instrument was chosen for its ease of deployment and industry verified capabilities. Each FDC1004 has four sensors and thus is attached to four sensing pads. A total of 52 FDC1004 was deployed onto the prototype board to allow for a combined sensor of 208 individual capacitance sensors. The green box in Figure 3.4 outlines all the FDC1004 sensors.

Due to the high number of sensors needed and used, the volume of data generated from all the sensors will naturally be very high. The ideal choice for handling these data would be the use of an FPGA, but an FPGA is somewhat cumbersome to program and would take an extended amount of time to debug. As the time to deployment is of most importance in the prototyping and testing stage, efficient hardware usage is not a significant priority in our case. Therefore, it was decided to use an STM-based ARM (STM32F7) based on its ease of debugging and programming.

Communication to and from the FDC1004 is done using the I2C protocol. Due to every FDC1004 having the same physical I2C address, 52 unique I2C channels would be required, meaning a minimum of 104 digital IO pairs would be necessary. This high I2C count presents a challenge to ARM MCUs as most of them do not have so many digital IO channels, especially that many I2C modules. Therefore, the design choice was to offload the I2C communication to smaller secondary MCUs (STM32G03). The smaller MCUs were responsible for communicating with the FDC1004, while the larger main ARM MCU (STM32F7) was focused on communication between the smaller MCU and the host PC. Figure 3.5 shows the communication link between each hardware step.

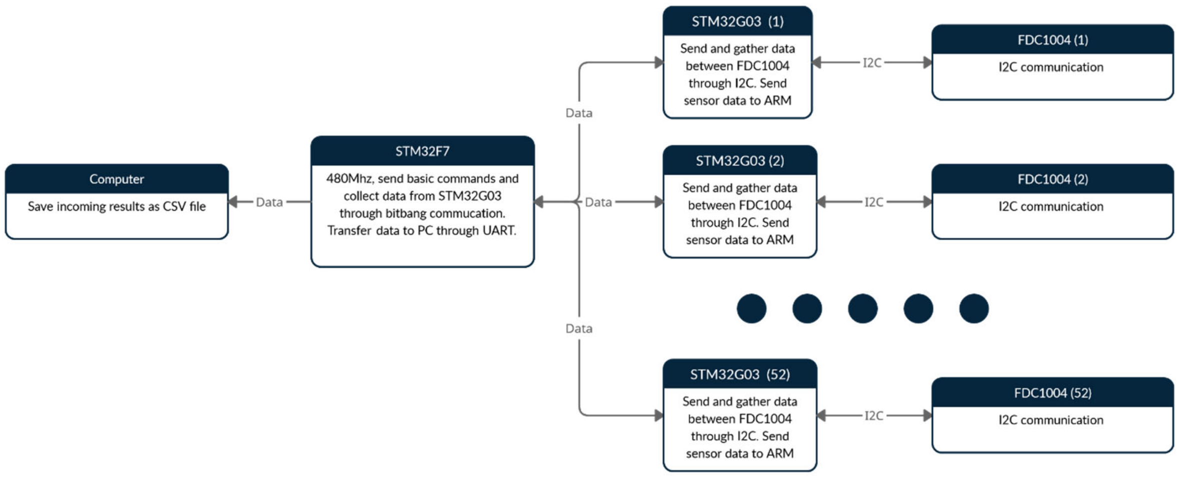

STM32G03 (1)
Send and gather data between FDC1004 through I2C. Send sensor data to ARM
FDC1004 (1)
I2C communication
I2C
Data
Computer
Save incoming results as CSV file
Data
STM32F7
480Mhz, send basic commands and collect data from STM32G03 through bitbang commucation. Transfer data to PC through UART.
Data
STM32G03 (2)
Send and gather data between FDC1004 through I2C. Send sensor data to ARM
I2C
FDC1004 (2)
I2C communication
Data
STM32G03 (52)
Send and gather data between FDC1004 through I2C. Send sensor data to ARM
I2C
FDC1004 (52)
I2C communication

Figure 3.5: *Hardware communication flowchart*

3.3.2 FDC1004 Design and Implementation

The FDC1004 represents the most critical component in our entire system as it performs the capacitance measurement. The FDC1004 is capable of performing capacitive sensing with grounded capacitor sensors and is designed specifically for implementing capacitive sensing solutions [21]. Each channel has a full-scale range of ±15 pF with a measurement resolution of 0.5 fF. Careful design and implementation are required to ensure the lowest possible noise and to ensure the operation of the combined 52 FDC1004 sensors do not cause significant crosstalk.

3.3.2.1 FDC1004 Communication Protocol

The FDC1004 is controlled and transmits capacitance value data through the I2C protocol [21]. While I2C was designed to have a single master device communicate with multiple slave devices on a single I2C channel, we cannot do this in our case as the FDC1004 only has a single factory hardwired I2C address. Therefore, we must be able to provide a total of 52 unique I2C channels dedicated solely for I2C communication for our usage case. Most MCUs available today can provide a small number of I2C communication channels, so we will need multiple MCU to achieve the I2C count. Yet most MCU today often require a large number of peripheral circuits to facilitate the MCU. For example, the MCU ATmega328 used on the Arduino Uno requires at least a clock oscillator for high-frequency operation and multiple by-pass capacitors[41]. This would increase the board space and component costs considerably. Therefore, we must choose an MCU with I2C with minimal peripheral circuits requirements in order to streamline the process of implementing the 52 FDC1004 I2C communication pairs.

The choice was made to use the STM32G03[42], which is available in an 8-pin package that fulfils our use case in the smallest footprint. The STM32G03 is a self-contained package, with its own onboard oscillator clocked at 16Mhz powered directly through Vcc and an internal PLL circuit, which produces a final clock frequency of 32 Mhz. The only peripheral circuits needed for the STM32G03 are three bypass capacitors for its 3.3V power supply. An additional two resistors are also required as per defined by the I2C communication protocol. While each STM32G03 contains two I2C communication channels, we chose to use only one channel as other pins of the second I2C channel are occupied for other functions. Thus for 52 FDC1004, there are 52 STM32G03, creating 52 pairs of STM32G03 and FDC1004; each pair has its isolated channel of I2C communication.

While the above STM32G03 solution solves the I2C hardware address issue of the FDC1004, it also introduces the problem of how to facilitate data transfer from STM32G03 to the host PC. Due to the high number (52) of individual STM32G03, we cannot have them all communicate with the host PC directly over the USB protocol as USB only accepts one connection at a time unless a switch is used. Therefore, we must have some form of "middleman" to facilitate this data transfer between the STM32G03 and the host PC. As seen in Figure 3.6, the chosen method uses an STM32F7 Nucleo board. The STM32F7 Nucleo board will communicate and collect the FDC1004 capacitance value data from the 52 STM32G03, and then transfer said data to the host PC through the UART protocol. The PC will only have to communicate with a single MCU through this method.

However, the data transfer method between all the STM32G03 and STM32F7 will also require a high number of digital IO. Even with only 2 IO used for each STM32G03, it will require a total of at least 104 digital IO dedicated purely for communication purposes between STM32G03 and STM32F7. The use of 104 digital IO is not feasible as we do not have so many available empty digital IO on the STM32F7. Therefore, the communication between STM32G03 and STM32F7 is a custom bitbang shown in Figure 3.6.

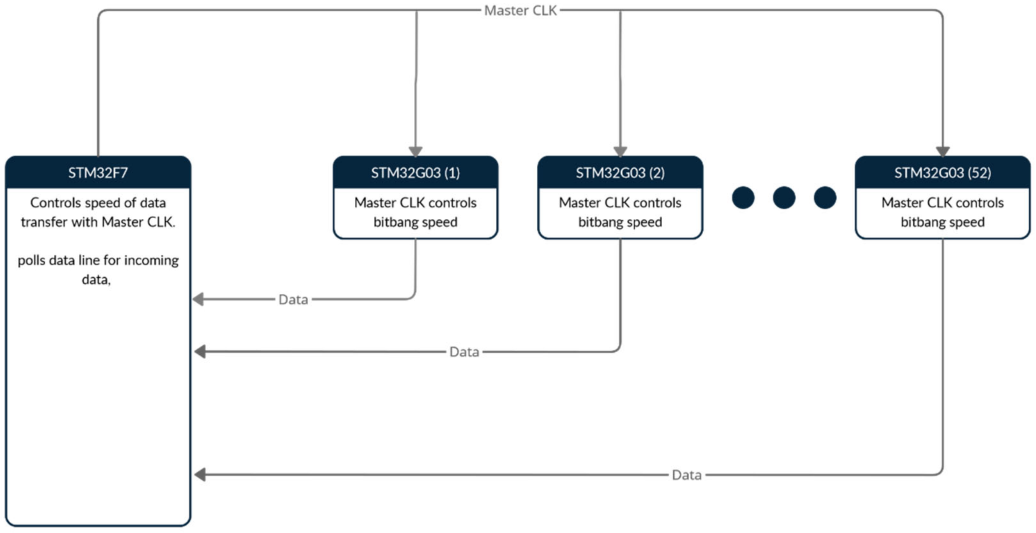

Master CLK
STM32F7
Controls speed of data transfer with Master CLK.
polls data line for incoming data,
STM32G03 (1)
Master CLK controls bitbang speed
STM32G03 (2)
Master CLK controls bitbang speed
STM32G03 (52)
Master CLK controls bitbang speed
Data
Data
Data

Figure 3.6: *STM32F7 communication with STM32G03*

Each STM32G03 will have one unique dedicated IO while also sharing one master IO with all other STM32G03. Data bits are controlled by STM32G03 and read by STM32F7. Master CLK is controlled by STM32F7 and read-only by STM32G03.

The master CLK acts as the master communication clock for all STM32G03, and the STM32F7 directly controls this CLK. At the start of the data transfer cycle, the STM32F7 will request the STM32G03 to send the first bit by turning master CLK low to high and back low again. This transition signals to the STM32G03 to send the first data bit. Once the STM32F7 has finished reading all the STM32G03 data, it will turn master CLK low to high and back low again, which tells STM32G03 to send in the next data bit. STM32G03 will stretch and hold a bit indefinitely unless signalled by the Master CLK to transmit the next bit or if a timeout of 5ms has occurred. For example, if the current bit is 1, the STM32G03 will hold the Dataline IO high until it has received the signal on the master CLK for the next bit.

Through this custom bitbang method, we can reduce the number of required digital IO from the previous 104 to 53 consisting of 52 data IO and 1 Master CLK IO. Nevertheless, this method of communication comes with its downside. No communication is verifiable, and any corrupt bits are sent and interpreted as is. The timing of the bitbang is also extremely delicate and requires tedious trial and error and fine-tuning as the clock speeds of the STM32G03 and STM32F7 are different.

3.3.2.2 FDC1004 to Connector Routing

Once the software for the FDC1004 is sorted the final part is to physically route the FDC1004 sensors to the SAMTEC connector, which will then be connected to the sensor board. With respect to Figure 3.4, this is presented as the connection between Green and Yellow. Each FDC1004 has four capacitance sensors, CIN1 to CIN4, and each FDC1004 also has its own unique shield trace generator; this is shown in Figure 3.7.

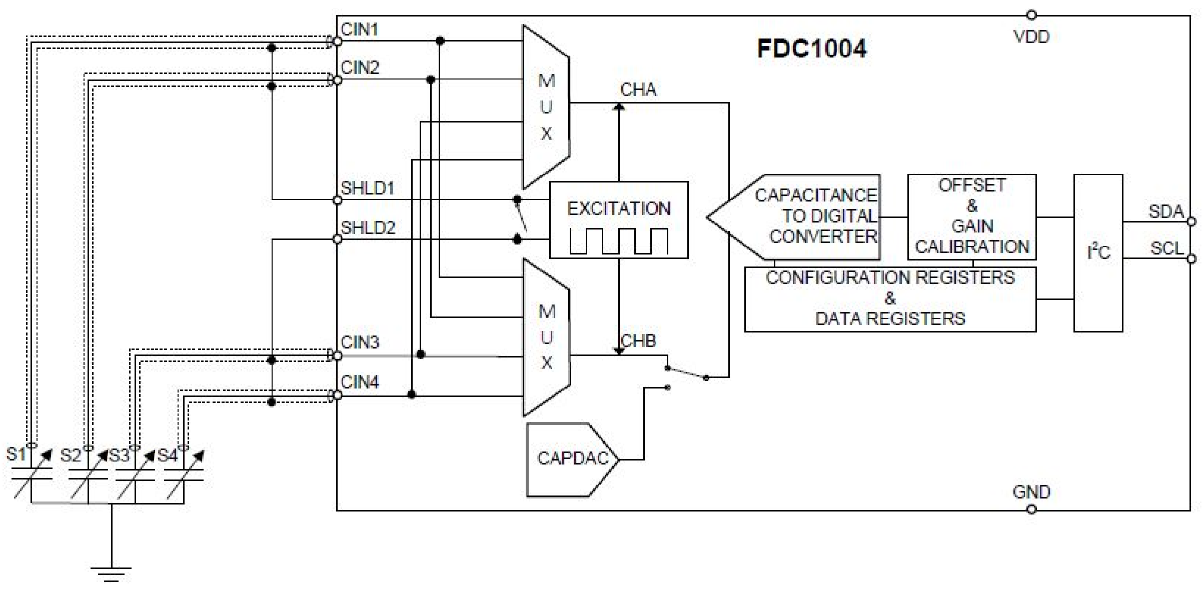

CIN1
CIN2
SHLD1
SHLD2
CIN3
CIN4
FDC1004
VDD
GND
M U X
CHA
CHB
EXCITATION
CAPACITANCE TO DIGITAL CONVERTER
OFFSET & GAIN CALIBRATION
CONFIGURATION REGISTERS & DATA REGISTERS
I²C
SDA
SCL
CAPDAC
S1
S2
S3
S4

Figure 3.7: *FDC1004 Capacitance sensor Pinout, Courtesy of Texas Instruments [21]*

Routing these CIN1-4 and the shield traces (SHLD1, SHLD2) is critical as all routed traces on the PCB are analogue signals, and any noise picked up will directly impact the capacitance value accuracy. Therefore, it is paramount to ensure that each capacitance sensor (referred to as CIN) will receive the highest possible shielding from external interference.

To minimize interference from the surrounding environment on the CIN, a thin PCB trace (6 mils) along with interweaved routing was chosen. Thin PCB trace reduces both stray capacitances picked up from the ground plane on the PCB and also EMI from the surrounding environment. Interweaved routing, where a shield trace is routed between every CIN trace, allows for the maximum isolation between different CIN, so coupling between two adjacent CIN is reduced to the minimum. Figure 3.8 shows the routing method used; red denotes the CIN traces, while yellow denotes the shield traces. In this design, no CIN trace is directly next to another CIN trace.

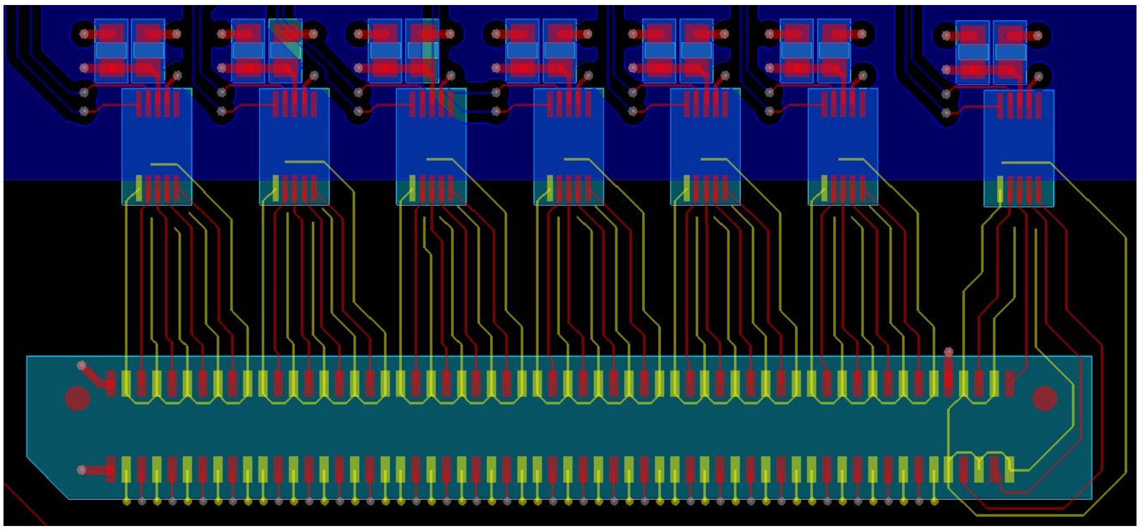


Figure 3.8: *Interweaved routing of Cin and Shield on the top side of PCB, Yellow is the Shield, and red is the CIN. Similar routing is performed on the bottom side of the PCB (not shown in this figure)*

The connector used to connect the Motherboard to the sensor board is the EdgeRate series connector by SAMTEC[43], shown in Figure 3.8 in teal and comes in male and female variants. The female version was used on the Motherboard, while the male was used on the sensor board. There are 120 contacts per EdgeRate connector for 480 contacts total on the Motherboard. Each CIN would occupy one contact on the connector, meaning 208 contacts are used for CIN; the rest of the contacts were allocated to shield trace and ground. By having so many contacts dedicated to shield traces, we effectively allowed each CIN trace to be wrapped by shield traces, thereby following our previous design motto; no CIN trace is directly next to another CIN trace.

3.3.3 Motherboard Power Delivery

Once the basic designs of the critical components have been established, the next step is to deliver power to the Motherboard and power all the components. A noisy power supply can leak noise to IC and the FDC1004 and directly affect their performance. Therefore, it is paramount that proper supply regulation is achieved, where noise is kept to a minimum.

Table 1: *Power supply rail voltage and current*

Voltage	Designed Current	Max Current	Number of rails
5.0 V	1.5 A	3.0 A	1
4.1 V	1.0 A	2.5 A	1
3.3 V	100.0 mA	2.0 A	4

Table 1 outlines all the power supply rails voltage and current rating used in the design, Figure 3.9 shows their respective locations on the Motherboard. In total, the power supply design contains one buck regulator and five LDO (low voltage dropout regulator), The buck regulator is used to drop input voltage to suitable voltages for the LDOs. The LDO serves multiple purposes, primarily as voltage regulators to power components but also as a power supply filter to filter the buck regulator's output noise.

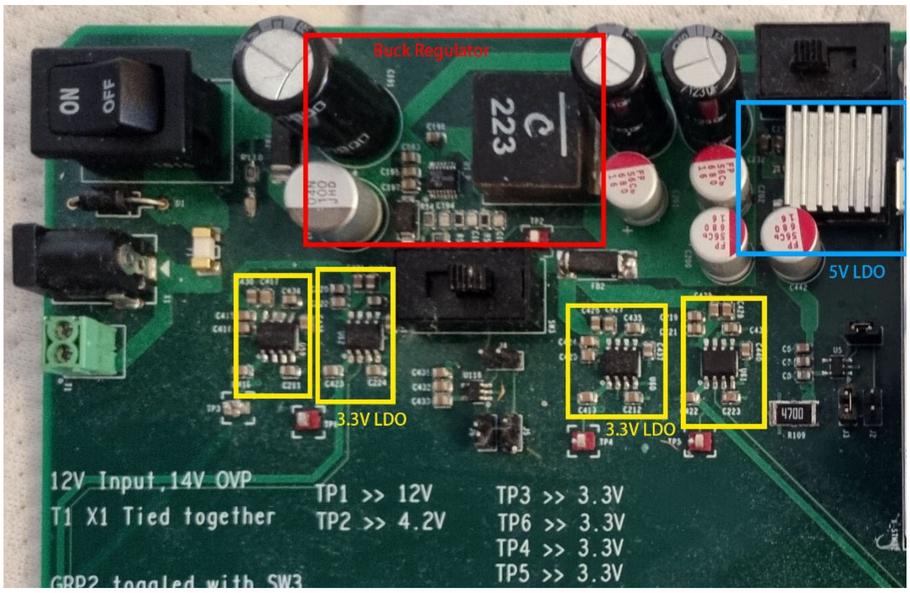


Figure 3.9: *Regulators on the Motherboard*

Figure 3.10 shows the voltage input and voltage output flow diagram and devices that the power respectively supplies. The 12 V input to the Motherboard through the 12 V barrel jack is connected directly to the 5 V LDO and the Buck regulator. The Buck regulator (outlined in red in Figure 3.9) serves the purpose of lowering the voltage to 4.1V, which then powers the 3.3 V LDO. Since converting 12 V to 3.3 V through LDO is not only highly inefficient and generates high amounts of heat waste but also degrades noise characteristic of the LDO due to the high difference between input and output voltage. 4.1 V is an ideal compromise between the safety margin of the 3.3 V LDO operation limits and heat generated.

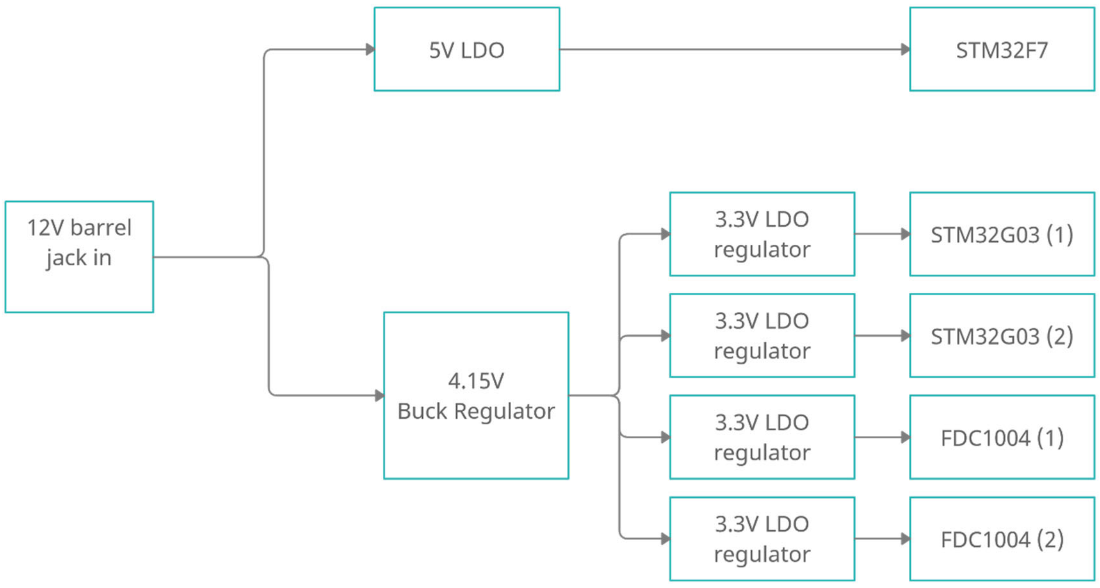


Figure 3.10: *Power flow diagram of Motherboard*

A 5 V rail is required to power the MCU, as the buck regulator can be noisy and outputs only 4.1 V. We opted to use a 5 V LDO that is sourced by the 12 V barrel jack input. The use of a heatsink was deemed necessary to maintain the LDO at cool-to-touch temperatures.

As stated, The MLB connects to external power through the 12 V barrel. The Buck regulator can accept a maximum of 15 V input voltage and therefore measures must be taken to ensure no overvoltage occurs. This is then followed by a fuse and a 14 V OVP Zener diode to ensure the input voltage never exceeds 14 V.

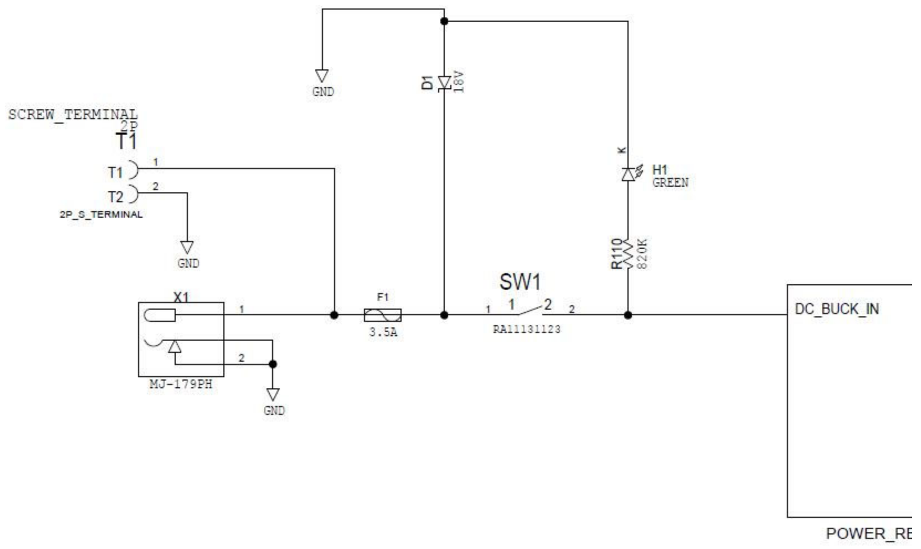


Figure 3.11: *Motherboard external power stage input design*

Figure 3.11 shows the simple protection circuit used in the case of overvoltage. Component X1 is the barrel jack and is tied to the T1 screw terminal for easy debugging, both are then connected to a fuse (F1) and a 14 V Zener diode (D1). If an input voltage exceeds 14V then the Zener diode will conduct, cause a short circuit and blow F1. The on/off switch is placed after the F1 fuse and D1 Zener diode combo, the rationale behind this is that this allows the circuit to detect an over-voltage presence regardless of the on/off state of SW1 and will allow the fuse (F1) to blow before the incorrect voltage is applied to the Buck regulator input. LED (H1) is used as an indicator to see if SW1 is closed.

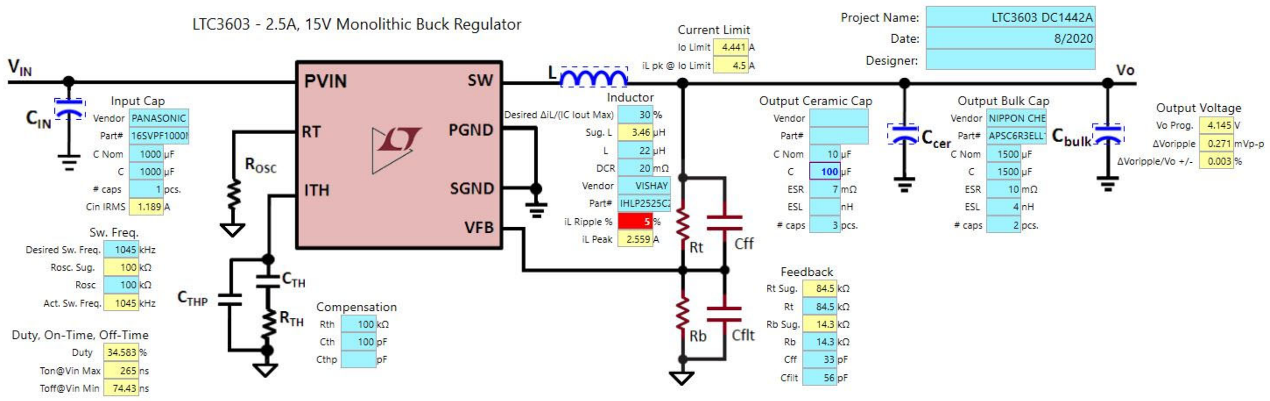
3.3.3.1 Buck Regulator

While a Buck regulator can effectively drop voltages without significant energy loss, it naturally presents the problem of emitting switching noise that is radiated to the surroundings and transmitted through the FR-4 dielectric of the PCB to nearby components. This noise is undesirable as it can be picked up by the FDC1004 that translates to noise on our final measurements. This noise is further compounded by our sensor boards exposed copper pads, which can also pick up this emitted switching noise and other surrounding EMI. Therefore, it is paramount to suppress the noise generated by the Buck regulator to the minimum.

The chosen buck regulator chip is the Analog Devices LTC3603 [44]. This device satisfies most of our requirements, with built-in dual FET and synchronous operation and a peak output of 2.5 A. Implementation of this Buck regulator is straightforward and provides a simple design phrase. However, more importantly, simulation is available through the Analog Devices LTpowerCAD software [45]. This ease of implementation ensures a fast and reliable Buck design that is easily operational verified.

The LTC3603 can be operated in both burst or continuous operation, and the max operation frequency is at 1 Mhz. For our design, we used Continuous mode with a 1 Mhz operation frequency. 1 Mhz ensures quick transient response and continuous mode ensures the least amount of interference to surrounding as the LTC3603 generates no start/stop phase noise. To address any possible radio frequency noise from sharp rise/fall edges from the 1Mhz operation, we use a ferrite bead at the Buck Regulator's output to reduce such effects.

Buck regulator being a feedback network, also requires simulation to ensure the system's stability, this simulation is done with the previous mentioned LTpowerCAD software. Figure 3.12 shows all the component choices used for the LTC3603 in designing a stable Buck regulator.



LTC3603 - 2.5A, 15V Monolithic Buck Regulator

Project Name:	LTC3603 DC1442A
Date:	8/2020
Designer:	

Input Cap

Vendor	PANASONIC	
Part#	16SVPF1000I	
C Nom	1000	µF
C	1000	µF
# caps	1	pcs.
Cin IRMS	1.189	A

Sw. Freq.

Desired Sw. Freq.	1045	kHz
Rosc. Sug.	100	kΩ
Rosc	100	kΩ
Act. Sw. Freq.	1045	kHz

Duty, On-Time, Off-Time

Duty	34.583	%
Ton@Vin Max	265	ns
Toff@Vin Min	74.43	ns

Compensation

Rth	100	kΩ
Cth	100	pF
Cthp		pF

Current Limit

Io Limit	4.441	A
iL pk @ Io Limit	4.5	A

Inductor

Desired ΔiL/(IC Iout Max)	30	%
Sug. L	3.46	µH
L	22	µH
DCR	20	mΩ
Vendor	VISHAY	
Part#	IHLP2525CZ	
iL Ripple %	5	%
iL Peak	2.559	A

Output Ceramic Cap

Vendor		
Part#		
C Nom	10	µF
C	100	µF
ESR	7	mΩ
ESL		nH
# caps	3	pcs.

Output Bulk Cap

Vendor	NIPPON CHE	
Part#	APSC6R3ELL	
C Nom	1500	µF
C	1500	µF
ESR	10	mΩ
ESL	4	nH
# caps	2	pcs.

Output Voltage

Vo Prog.	4.145	V
ΔVoripple	0.271	mVp-p
ΔVoripple/Vo +/-	0.003	%

Feedback

Rt Sug.	84.5	kΩ
Rt	84.5	kΩ
Rb Sug.	14.3	kΩ
Rb	14.3	kΩ
Cff	33	pF
Cflt	56	pF

Figure 3.12: *LTC3603 Surrounding circuit*

The capacitor and inductor choices are essential as they directly affect the feedback stability and ripple size. Thus the choice of large inductor and output capacitors ensures a low ripple at the output of the buck regulator. This low ripple would directly affect the 3.3 V LDO that comes after the buck regulator as while the LDO has a high power supply rejection ratio (PSSR), it is nevertheless not infinite. Therefore, the lower the output ripple of the Buck regulator, the better the performance of the LDO would be. The Loop response of the Buck is also of importance as unstable buck regulators can swing widely when transient loads are presented, which will be the case when MCU is connected that constantly switches between high current and low current draws. Therefore, it is crucial to ensure sufficient simulation is performed on the loop stability.

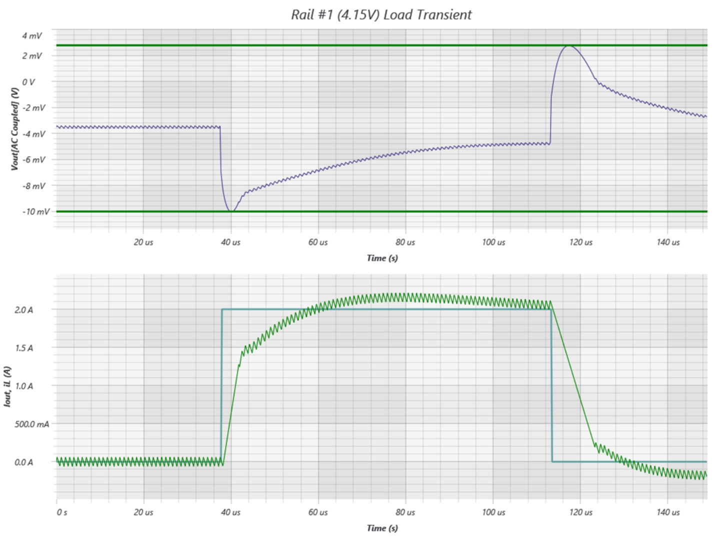


Figure 3.13: *LTC3603 Load transient 2A load*

Figure 3.13 shows the time-based transient load simulation performed by Analog Device's LTpowerCAD software. The purple line above is the simulated transient of the buck regulator when a transient load of 2.0 A is drawn from the buck regulator, and the green line in the second graph is the simulated current output. As the simulation shows, even in the extreme case of 2.0 A transient load, the output voltage does not dip below 10.0 mV or rise above 4.0 mV of the target voltage of 4.15 V and the recovery time is approximately 100.0 µs. This severe case of 2.0 A load transient should not happen in our use case unless faults such as short-circuit occur. If shorting faults do occur, ideally, the fuse should react in time to stop damages to the LTC3603.

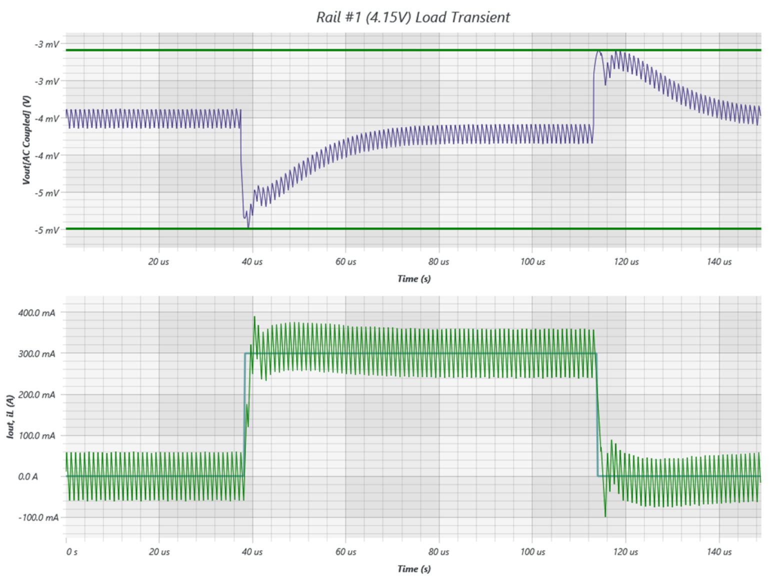


Figure 3.14: *LTC3603 load transient 300.0 mA load*

Figure 3.14 shows a much more realistic load transient where the transient is 300.0 mA. In such a case, the voltage should only dip a maximum of 5.0 mV below nominal 4.15 V, and the recovery time is shortened to approximately 60.0 µs.

The Bode plot of the buck regulator shown in Figure 3.15 agrees with the graphs in Figure 3.13 and Figure 3.14 in that the Buck regulator is indeed stable. With the component selection shown in Figure 3.12, we can achieve a phase margin of 130.0 degrees and a gain bandwidth of 31.0 kHz. The phase margin is much higher than the recommended 60.0 degrees. However, the bandwidth ideally can be higher than 31.0 kHz, but it is nevertheless sufficient. From Figure 3.15, we can also see that after the 1.0 MHz frequency, the LTC3603 would rapidly become unstable, which agrees with the maximum operating frequency of the LTC3603 from its datasheet.

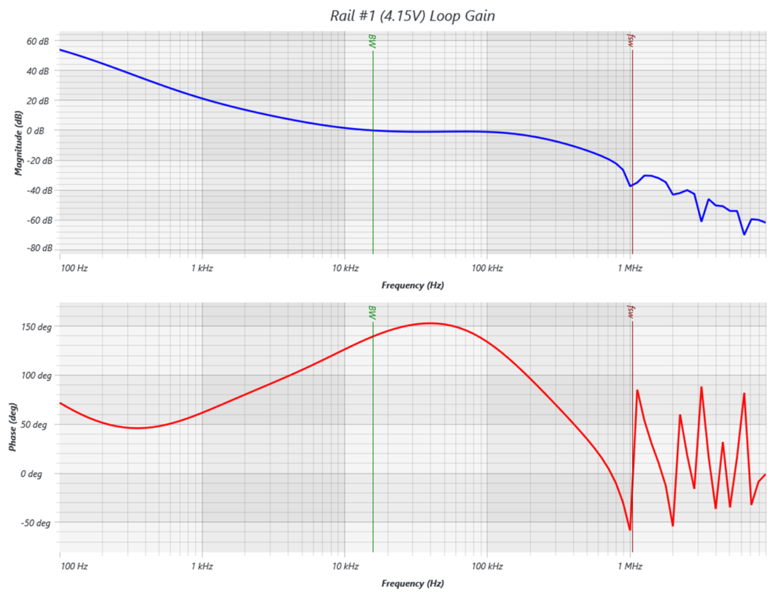


Figure 3.15: *LTC3603 Bode plot*

3.3.3.2 Linear Dropout Regulator (LDO)

The purpose of using LDO instead of making the Buck regulator output 3.2 V and directly using it to power the MCU and FDC1004 is that both FDC1004 and STM32 are susceptible to noise on their respective input voltage. The noise from the buck regulator and any noise from MCU's internal clock could affect the FDC1004 if the MCU and FDC1004 are both directly powered by the buck regulator. The FDC1004 has a published PSRR of 13.6 fF/V, which means 1.0 volt of noise can cause 13.6 femto-farad of measurement error.

As the FDC1004 is affected by power supply ripple, all reasonable design choices must be taken to accommodate this. Therefore, for powering the FDC1004 and their respective STM32G03, the chosen LDO was the Analog Devices ADP7158 [46]. The ADP7158 has extremely high PSRR at 60.0 dB at the 1.0 MHz input frequency range when the load is under 1.2 A [46] and an output noise level of 1.6 µV RMS at max current level [46]. Therefore, this chip is suitable for use cases where it can sufficiently suppress and filter the switching noise from the Buck regulator and provide clean DC into the FDC1004.

Figure 3.16 gives a graphic overview of which separate devices are powered by which individual LDO. The Yellow and green in Figure 3.16 are LDOs for the FDC1004, white and red are for the STM32G03. At the same time, the blue/teal is the LDO for the STM32F7 Nucleo board.

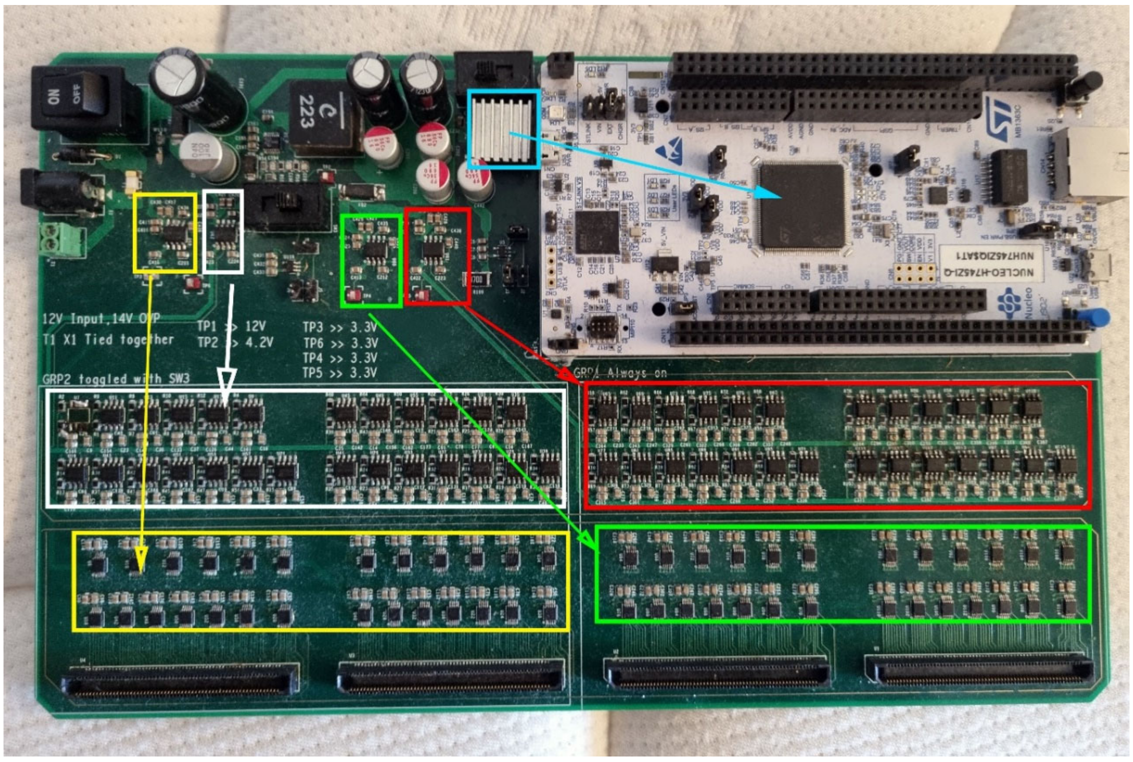


Figure 3.16: *Components powered by their respective LDOs*

The purpose of using different LDOs from the STM32G03 and the FDC1004 is that since as shown in Figure 3.5, each FDC1004 has its unique STM32G03 that performs the I2C communications, and since the STM32G03 is an MCU, naturally high frequencies are involved as the STM32G03 has internal PLL and buck regulators [42]. To reasonably isolate these issues, the decision was taken to split the LDO between the STM32G03 and the FDC1004. The decision to further separate the STM32G03 and FDC1004 into two groups (white and red), (yellow and green) shown in Figure 3.16 is to allow for debugging and failsafe considerations as if a catastrophic fault occurs, at least half the Motherboard would remain usable.

The LDO used to power the STM32F7 would require a significant size to accommodate the heat generated by the LDO due to the high current requirement of the STM32F7. The regular operation of the STM32F7 Nucleo board requires 500.0 mA of current at 5.0 V [47]. As the LDO converts 12.0 V to 5.0 V at 500.0 mA, this means the LDO must be able to dissipate 3.5 watts of heat in the worst-case scenario. Following these requirements, the device NCP59150 [48] by On Semiconductor was chosen. It comes in a D2PAK3 package and is capable of a 1.5 A supply current but, more importantly, has a thermal rating of R_JA = 52ºC/W when used as Junction-to-air, which is lower than most LDO that comes in smaller packages. When supplied with a small passive heatsink, the R_JA is further reduced to a much smaller value; the temperature observed in our use case is approximately 60°C and average current draw of 350.0 mA, which gives approximately R_JA = 24ºC/W. Noise and voltage regulation is not as necessary of a factor as the STM32F3 has onboard voltage detection and safety cut-offs [47]. Nevertheless, NCP59150 has a PSRR rating of 30dB at 200Hz and an output accuracy of 2.5 %. The noise of this LDO is not extremely important as the STM32F7 is a digital chip and has its own internal LDO, either way, no noise figure was supplied by ON Semiconductor for NCP59150.

3.4 Sensor-pad Board Design

The physical board is shown in Figure 3.17. The sensor pads are responsible for sensing the variation in capacitance. The design of this sensor board and sensor pad directly relates to what is captured by the FDC1004.

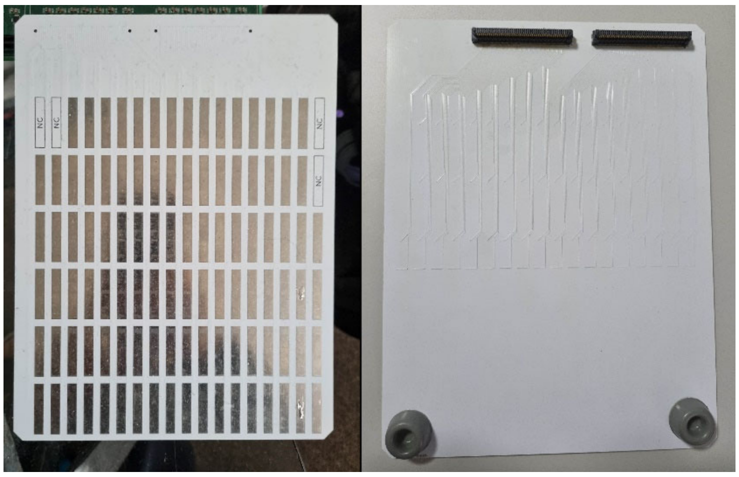


Figure 3.17: *Proposed Sensor board design*

Therefore, close attention must be paid to its layout plan to ensure optimal performance. The larger the sensor "pads" (silver rectangles in left of Figure 3.17), the better the sensitivity and spatial resolution. But as a hand is only so big, the larger the sensor the smaller the overall resolution/pixel of the final capacitance "image". Thus, we must find other means of increasing spatial resolution to the maximum for a given pad size. Ultimately a trade-off between size, sensitivity, noise immunity and coupling effects must be made [49].

3.4.1 Noise, Sensitivity, and Shielding

As stated before, the biggest issue we faced in the design of the system is the noise picked up by the FDC1004. Therefore, we must attempt to keep this noise to its minimum for our sensor pad design. The most direct method is by increasing the separation of two pads; however, this would again reduce the overall pixel of the capacitance "image", and thus we must employ other techniques to best increase the performance of the sensor pads.

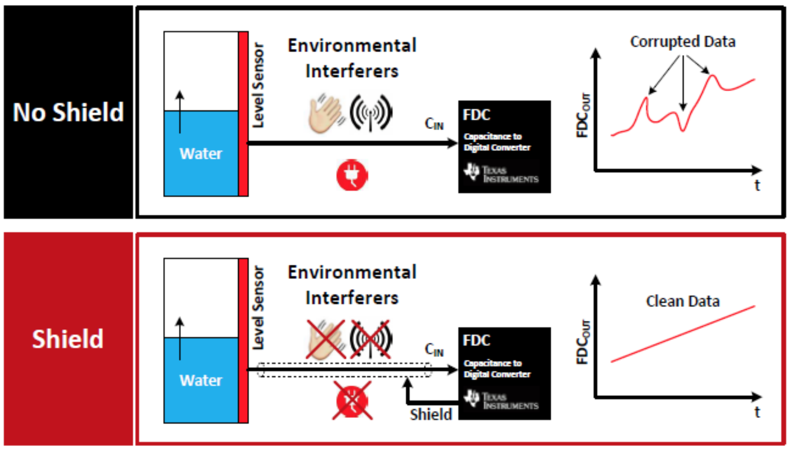


Figure 3.18: *Sensor shielding, Courtesy of Texas Instruments [49]*

As shown in Figure 3.18, one of the significant sources of interference and noise in capacitive sensing is parasitic capacitance to the ground along the signal path between the sensing device trace and the electrode sensor. This parasitic noise can cause unintended alteration in the final reading of capacitive sensors. Worst yet, this alteration can vary from sensor to sensor and trace to trace. The shield driver on the FDC1004 attempts to reduce this issue by using an active signal output that is driven at the same voltage potential as the sensor input. So ideally, there is no potential difference between C_{in} (capacitance sensing pin) and its surroundings, and thus no current leakage can occur. Multiple different methods of routing this shield trace can be implemented, and each will have its trade-offs, Figure 3.19 shows the four common methods.

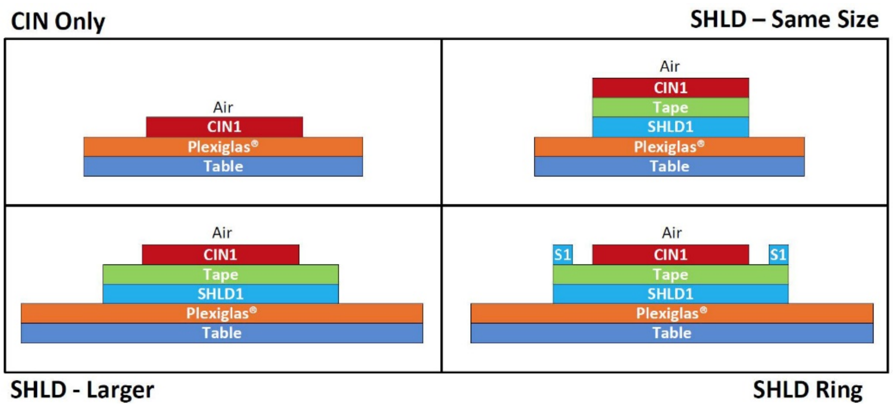


Figure 3.19: *methods of shield routing: no shield, shield under but the same size as sensor pad, shield larger than CIN and both Shield under along with a shield ring around the sensor pad, Courtesy of Texas Instruments [49]*

Clockwise, each shielding type shown in Figure 3.19 offers increasing resistance to stray capacitance yet decreasing sensitivity and spatial resolution. Results published by Texas instruments on the expected decrease in sensitivity of a single sensor are given in Figure 3.20. The sensor pad size in Figure 3.20 is 20.0 mm by 20.0 mm; in comparison, the sensor we used was 5.0 mm by 27.0 mm, representing a 77 % reduction in surface area.

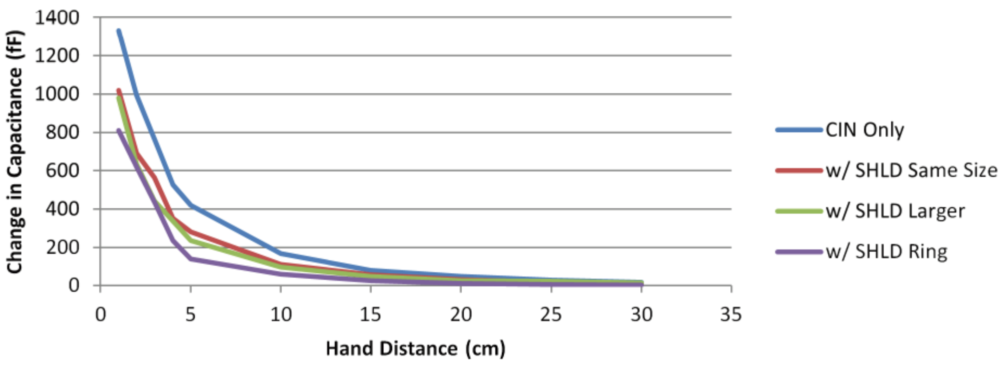


Figure 3.20: *Sensitivity of sensor relative to shielding type, Courtesy of Texas Instruments [49]*

From Figure 3.20, we can expect no shielding to have the best possible sensitivity. If we can account for the increase in noise through software means, no shield would be the best design.

Nevertheless, in reality, this noise is much too significant and random to be compensated with software means, and we must take steps to mitigate it through physical design means.

The best shielding design is the lower left and right design in Figure 3.19, which consists of shield traces all around the C_{in} trace. However, in our use case, we are not performing precision analogue measurements; the benefit of increased noise immunity does not outweigh the cost of decreased sensitivity. Figure 3.20 shows that one can expect a 500.0 fF drop in sensitivity when going from full shield to no shield when the sensor pad is 20.0 mm by 20.0 mm. The decrease of 500.0 fF in sensitivity is not viable for our use case. Therefore, we would use a reduced design compared to Texas Instruments; we would employ a shield trace around the sensor pad with no shield under it. Our design diagram is shown in Figure 3.21.

3.4.2 Final Design of Sensor Pad

We first established the baseline noise generated by sensor board designs. In the case of the no shield under the CIN (top left of Figure 3.19), the generated random noise is averaged to be 3253 (FDC1004 reading) or approximately 0.0062 pF over a 1.0 s period; the highest noise is 10445 or 0.0199pF. The results from the shields same size as the sensor pad (bottom left of Figure 3.19) gives the result of 3103 average noise or 0.0059pF, with the highest noise is observed at 5817 or 0.0111 pF. Finally, the design of the shield trace around the C_{in} trace (Figure 3.21) gives the result of 3152 average noise or 0.0060 pF, and the highest noise is observed at 7000 or 0.0133 pF. These results show that while the average is not decreased between the different designs, the peak/spike is greatly reduced. This finding agrees with Texas Instruments' findings that a random high spike of noise is massively reduced with a shield trace [49]; however, the higher the immunity to random spike, the higher the reduction in sensitivity.

In our experiment between our design (Figure 3.21) and fully shielded (bottom left of Figure 3.19), when we compared the detection range of the two designs, the fully shielded cases displayed a detection range of a mere 25.0 mm before no observable difference in reading could be made. For our design, a distance of up to 55.0 mm can be detected. However, it is worth noting that the data from the shielded sensor board has a much tighter spread of captured values, but the detected capacitance values drop off more quickly. In our design, while we can observe capacitance values up to 55mm, the capacitance reading varies more widely between values.

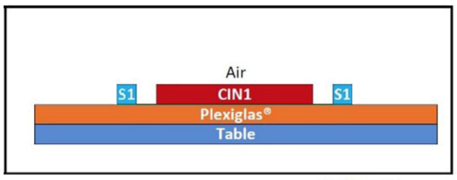


Figure 3.21: *our shielded sensor board design*

In our use case, we must base our decisions on range consideration as a range of 25.0 mm is simply not sufficient. Furthermore, software-based calibration may compensate for some of the increased variations in noise. Therefore, we opted to use the shielding design shown in Figure 3.21 as our design where we "wrap" sensor pads with shield traces.

3.4.3 Sensor Board Connection to Motherboard

As mentioned in section 3.3.2.2 , to enable the connection between the Motherboard and the sensor pad board, we opted to use the EdgeRate connectors by Samtec [43]. The left side of Figure 3.22 shows the sensor pad board and the two male connectors (blue); The right side of Figure 3.22 shows the motherboard and the two female connectors (red).

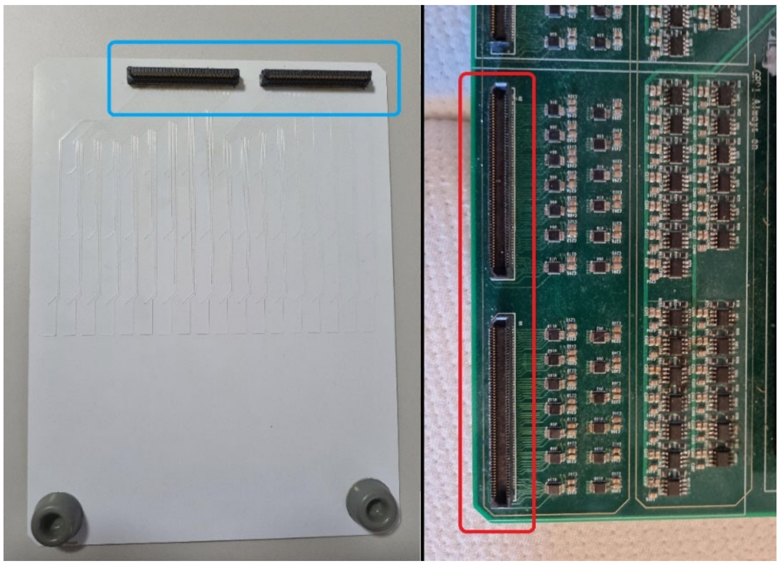


Figure 3.22: *Samtec Connectors, male connectors (blue) and female connectors (red)*

CHAPTER 4: SOFTWARE AND DATA COLLECTION

4.1 Introduction

The developed hardware outlined in chapter 3 was used to capture capacitance data from five test participants whom each performed five unique gestures multiple times. The experiments were conducted in a controlled environment where the most significant variable should be the difference in the participant's hand size and shape. The dataset was used to train, validate, and test several classifiers. We will first give an overview of the software and its operation parameters, followed by the data collection environment. Finally, we will examine the differences and variables between each dataset.

4.2 Types of Data Collected

Due to the iterative improvement of the hardware and software based on the findings of the experiments, a total of three sets of complete data were collected through the span of this study. Each dataset was collected after a major software revision.

The first two sets of data were collected under the assumption of evaluating the hardware and establishing the basis of all future data gathering. Dataset 1 served as a proof of concept that capacitive sensing-based gesture recognition was needed to justify further experiments into such a design. The second dataset was the intention of exposing software issues. Therefore, both the hardware and software aspects of the system were modified upon based on the results of Dataset 1 and Dataset 2. Dataset 3 utilised improved calibration and is used in all model evaluation and final accuracy comparisons. In its final form, five different hand gestures were captured and shown in Figure 4.1 for Dataset 3.

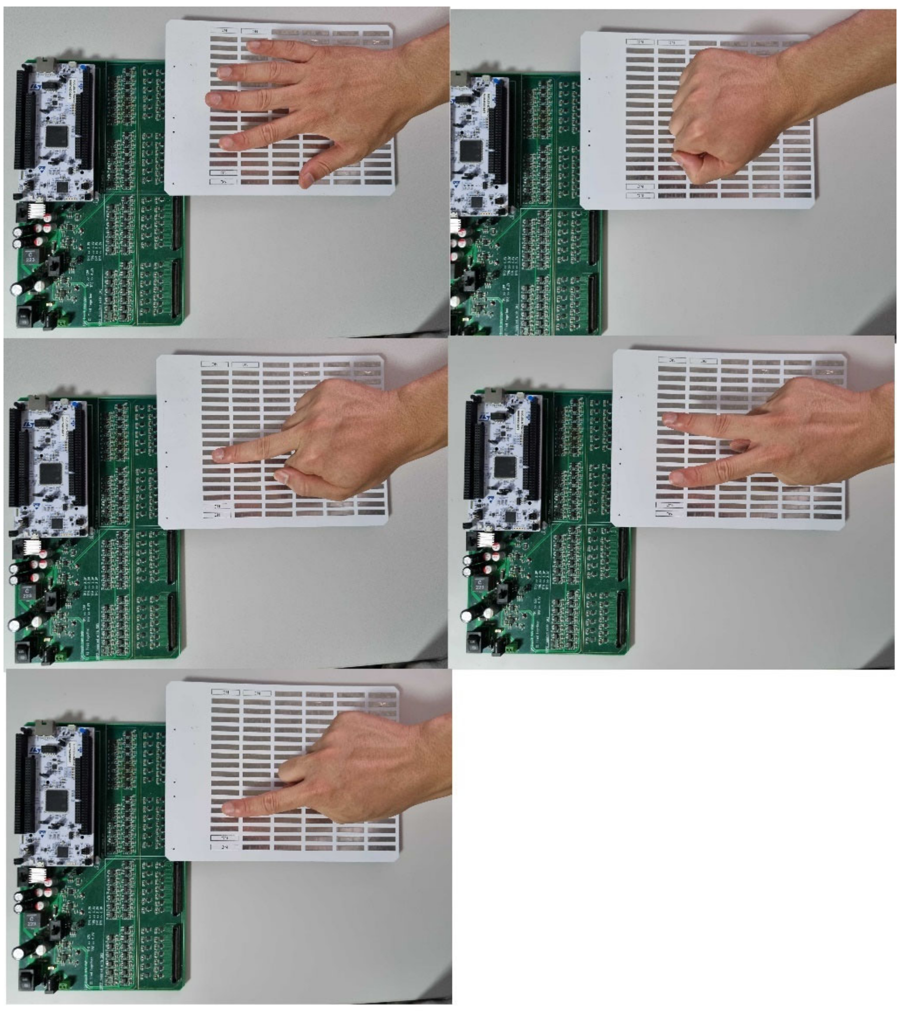


Figure 4.1: *All five hand gestures from top left clockwise: palm, fist, middle, ok, point. The gestures combine a mixture of similar and non-similar types to thoroughly test the classifier. For example, one would expect the classifier to have difficulty with the point and middle finger gestures*

4.3 Software implementation overview

The maximum measurable change in capacitance when a hand is 55.0 mm from the pad is the order of approximately 0.5 pF (please see 3.4.2). At this range, noise and environmental factors can dramatically affect readings. The FDC1004 IC chip requires generating an excitation signal that extends to the physical sensor pads when it actively measures capacitance. This excitation signal, unfortunately, is a potential noise source for any adjacent FDC1004 sensor pads. Therefore, running all FDC1004 concurrently is not sensible. Instead, a "sweep" is performed, where each FDC1004 is turned on, collects data one at a time for the four sensors and then turned off. After the first FDC1004 completes this cycle, the next FDC1004 would perform the same cycle until all sensors have completed this cycle. The STM32G03 would then combine the data from all four sensors to form a complete frame worth of data and send it to the STM32F7, which would combine all data from STM32F03 and send it to the host PC.

Each FDC1004 was set at 100 samples/s, and since each FDC1004 had four sensors connected internally through a multiplexer, it would take 40.0 ms to complete one entire cycle of sampling [21]. To account for start/stop commands, transfer time and sufficient gap between each FDC1004, each FDC1004 is allocated 50.0 ms for a complete sampling cycle of all four sensors and transferring data to the STM32F03.

4.3.1 Data Normalization

Due to the varying nature of the distance of each individual pad to the physical FDC1004 sensor, some form of data normalization must be implemented so that the different data collected from each pad are location-invariant and comparable to allow for easy debugging and neat presentation. The chosen method of mitigating said issue was using hardware programmed offsets outlined in the FDC1004 datasheet [21]. Each sensor within the FDC1004 was calibrated until they reached one Pico Farad. The value of one Pico Farad was chosen to allow a sufficient safety net for noise and drifts so as not to allow the sensor reading to dip below 0 Farad and into negative readings, which the software is incapable of reading. This calibration step is hardware-based, and each sensor must undergo calibration through a string of I2C commands. This calibration was a lengthy process that took approximately 5 seconds.

However, while the hardware can compensate the distance of each individual pad to the FDC1004, capacitive sensors have a tendency to naturally drift due to an array of other different reasons such as temperature, humidity and local interferences. When we measure down to the pico- and femto-farad range, this drift becomes an issue and must be actively controlled. The FDC1004 cannot account for this by hardware alone due to the slow speed of hardware calibration, and thus Texas Instruments actively encourage to use software calibration solutions for each use case[21].

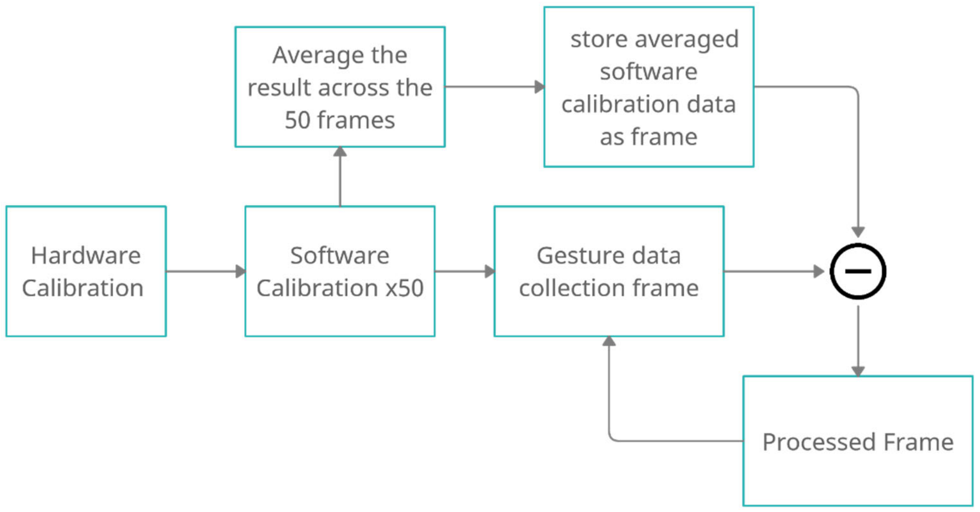

Average the result across the 50 frames
store averaged software calibration data as frame
Hardware Calibration
Software Calibration x50
Gesture data collection frame
Processed Frame

Figure 4.2: *50 frame Software calibration flow chart*

Figure 4.2 shows the software calibration process we used to perform offset and background subtraction was used to mitigate sensor drifting. The software calibration process is similar to hardware calibration, but due to not needing to write I2C cycles constantly, we can therefore increase the calibration speed. The operation starts the system by collecting capacitance values without a hand present. The values from this cycle are then stored into a matrix and named the calibration frame. Fifty consecutive calibration frames are collected and averaged to reduce the impact of environmental variables on the sensor pads. This averaged frame acts as the baseline empty pad values, and when we start collecting hand gestures, this baseline is subtracted from all data collected for the gestures. The use of the average of 50 frames of data was decided upon as a balance between speed and accuracy, as during test we found the use of only a single frame of data as the calibration frame did not yield satisfactory results.

4.3.2 Final Dataset: Dataset 3

Table 2 shows the composition of dataset 3 that was collected with the logic outlined in section 4.3.1 . All participants' data were collected on different days to allow for independent data between participants. The gesture is captured in the same manner as shown in Figure 4.1.

Table 2: *dataset three composition*

Number of unique participants	Number of gestures	Frames per gesture	Total frames per participant	Total number of frames for dataset 3
5	5	200	1000	5000

Even with the extensive software calibration used for data set three, there are still observable variations between data frames from different participants. This variation suggests that we can only assume data to be loosely correlated for the fairness of evaluating of a classifier. Data collected from two different environments cannot be considered to be similar even if collected on the same person. Moreover, if such an assumption is made, we cannot rely on the "traditional" method of mixing all data together and doing a 70/15/15 split for training, validation, and testing. Instead, testing data must be unique and separated from training and validation data. For example, participants 1-4 are used for training and validation only, while participant 5's data is used solely for testing purposes. This isolation of training/validation and testing data attempts to verify the robustness of the classifier when presented with previous untrained data when presented new data from a different day with random moisture and temperature. Later chapters of this thesis will show that splitting training and testing data is a more precise indicator of a classifier's capability to identify gestures performed by different individuals than doing the traditional 70/15/15 split.

CHAPTER 5: CLASSIFICATION MODELLING, DESIGN, AND RESULTS

5.1 Introduction

The collected hand gesture data from the five participants were split into training (and validation) set and test set to train and evaluate several multiclass classifiers. The classifiers include two probabilistic classifiers (Naïve Bayes and Decision Tree) and two neural networks namely multilayer perceptron (MLP) and convolutional neural network (CNN). For this thesis, TensorFlow 2.0 by Google is used on the python platform to implement the classifiers.

5.2 Classifier background

5.2.1 Probabilistic Classifier

The purpose of using the two classifiers is to evaluate how these simple models perform compared to the more complex neural networks.

5.2.1.1 Naïve Bayes Classifiers

The first of these networks to be tested will be Gaussian Naive Bayes, which remains one of the most effective and efficient classification algorithms [50]. Naïve Bayes Classifiers always assume that all features are independent of other features. This assumption makes it possible to achieve a high model accuracy with a relatively small training dataset.

Decision Tree Classifier

A decision tree usually consists of a root node, a number of interior nodes and finally, a number of terminal nodes [51]. The building block of a decision tree is recursive partitioning, which involves splitting the data into partitions or subsets. This process repeats itself until the partitions are sufficiently homogenous. In a simple sense, a decision tree asks simple yes or no questions until the model can determine which terminal node it wants to put the input data into. This process is extremely fast to perform and has a high accuracy for simple datasets when the data are sufficiently similar.

5.2.2 Neural Networks

The rise of neural networks began in the early 2000s when the increase of cheap graphical processing power and "deep" neural networks [52] made more complex neural networks possible. Before this, neural networks did not process the capability to surpass more traditional approaches due to the high processing power requirement, except for a select number of specialized cases. With today's processing power, the development of neural works rapidly flourished. This study mainly explores two Neural networks, Multi-Layer Perceptron (MLP) and Convolution Neural Network (CNN). While Both types of neural networks are tested, the main focus will be on multi-layer perceptron as we did not find evidence that convolutional neural networks produced better results than multi-layer perceptron in our use case.

5.2.2.1 Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron (MLP) is one of the most widely used Neural Networks that has seen success in a wide range of problems [53]. Figure 27 shows an example of such a model.

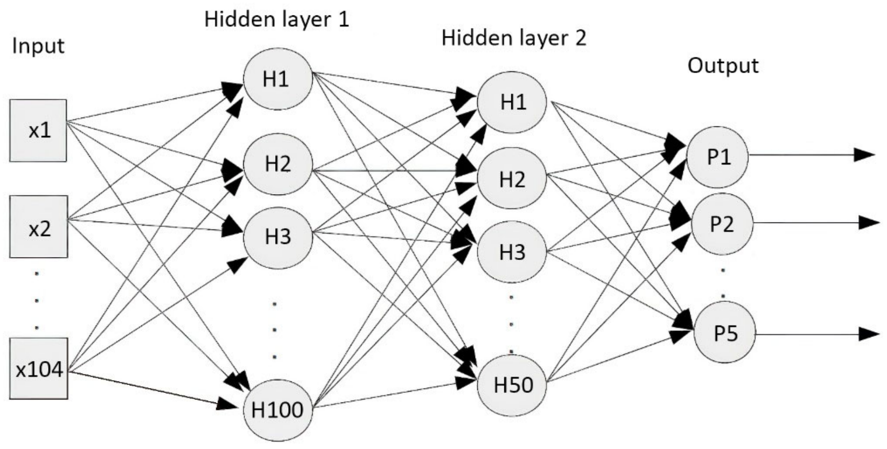


Figure 5.1: *Simple MLP model with two hidden layers [54]*

The MLP neural network is a feedforward network and consists of multiple layers of nodes. A complete MLP system consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. There is no limit to the amount of hidden layer that a can employ; more hidden layer means more nodes are possible, and therefore more paths are possible, albeit at the cost of higher computing cost.

Except for the input layer, where it receives its data through external means, all other layer receives their data from the previous layer. Once data is received, a node will use a nonlinear activation function to combine the input to the node with predetermined weights and then adding a bias to produce an output. A range of different activation functions exists.

The process of training an MLP model involves continuously adjusting the weights of the connection between nodes. This adjustment is based on the error in output (the difference between the expected result and the actual output). This continuous adjustment of the weights is a supervised learning process called "backpropagation" [55]. The backpropagation algorithm consists of two parts: the forward pass and the backward pass. In the forward pass, the expected output corresponding to the given inputs are evaluated. In the backwards pass, partial derivatives of the cost function with respect to the different parameters are propagated back through the network. The process continues until the error is at the lowest value [55].

5.2.2.2 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) distinguishes itself from other neural networks with superior performance in regard to image and audio signal inputs. Figure 5.2 shows an example of a typical CNN network.

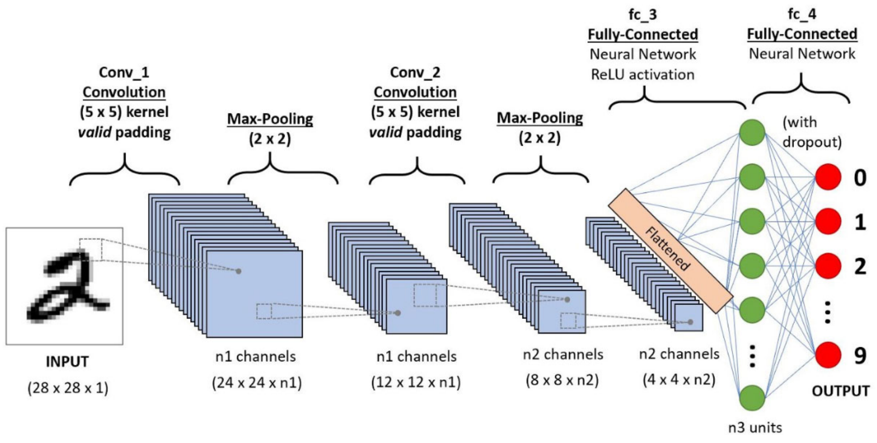


Figure 5.2: *typical CNN model [56]*

A typical CNN network consists of at least three basic layers: a convolutional layer, a pooling layer, and a fully connected layer.

The convolutional layer is the first layer of a convolutional network. The job of the convolutional layer is to reduce the images into forms easier to process without losing features. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer must be the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image [11] at the cost of more computational power.

The convolutional layer is the core of a functional CNN. It requires a few components, which are input data, a filter, and a feature map [57]. A feature detector, also known as a kernel or a filter, will move across the receptive fields of the image in blocks, checking if a feature is present. Usually, multiple filters will be used as a single filter isn't capable of capturing all the information, this process is known as a convolution. This process is usually performed on one colour at a time. An RGB image which would consist of a matrix of RBG, as seen in Figure 5.3, would require this process to be run three times.

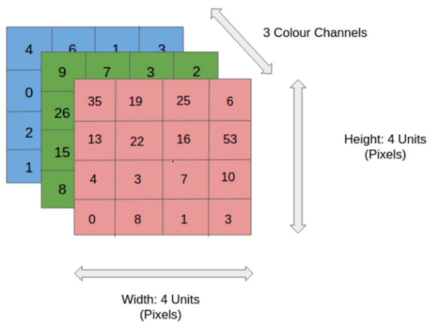


***Figure 5.3:** matrix of RGB image [56]*

The Pooling layers, also known as downsampling, conducts dimensionality reduction, reducing the number of parameters in the input. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights [57]. Finally, the fully-connected layer performs the job of actual classification.

5.2.2.3 Singular Value Decomposition

Singular Value Decomposition (SVD) has many applications in image processing; one of which is noise filtering [58]. SVD is a mathematical procedure to decompose a matrix into three matrices, which can be rewritten as a sum of rank-one matrices [59]. SVD attempts to sort data into multiple ”modes”, where hopefully the lower SVD modes should remain relatively clean, which suggests the possibility of data filtering by retaining only the lower modes [60].

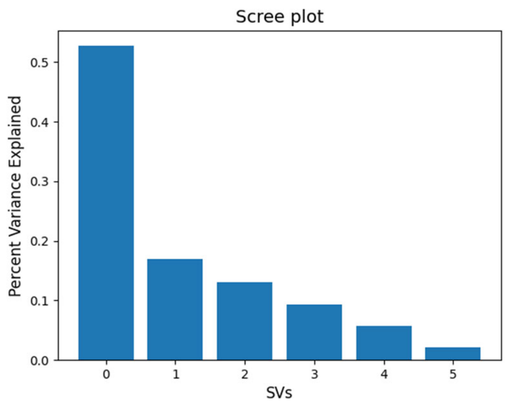


Figure 5.4: *Example of a single data frames SVD scree plot*

Figure 5.4 shows a typical scree plot of a single data frame from test participant one in the dataset. It can be seen that mode one contains around 50 % of the variance. We then assume mode zero contains most of the noise and reconstruct the matrix without mode zero. The fact that some valuable data may be removed in this process is of no concern if the final accuracy can be improved.

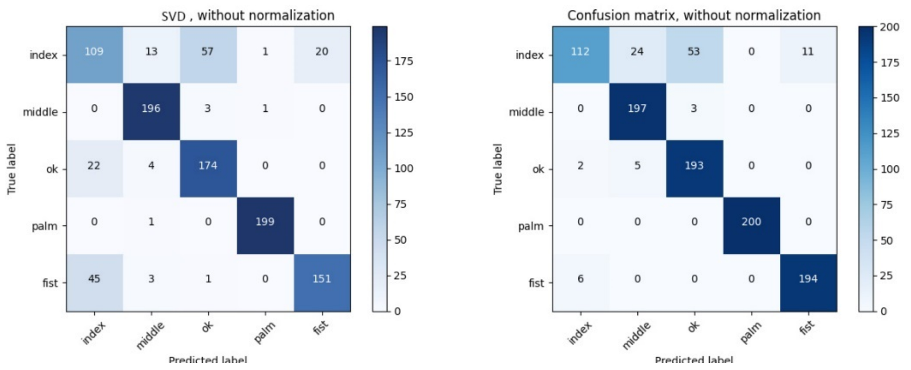


Figure 5.5: *SVD vs non-SVD results with training/validation with participant 2,3,4,5 test with participant 1*

Figure 5.5 shows the results used for evaluating SVD, the test is performed with participant 2, 3, 4, and 5 as training/validation and participant 1. Results of SVD shows a model accuracy of 82.9 %, without SVD we can achieve an accuracy of 89.2 %. This drop in overall accuracy suggests that some data in mode zero were removed and that it affected the final accuracy.

The conclusion from this is that while SVD can be used as a form of noise removal in some cases, we did not find it to offer a significant accuracy increase compared to results that did not perform SVD. Therefore, we opted not to perform SVD in our use case as it did not provide tangible benefits.

5.3 Probabilistic Classifiers Results

The first models to be tested are the probabilistic classifiers. The first test is with data from test participants 1-5 and mixing them together, then randomly splitting 75% for training/validation and 25% for testing.

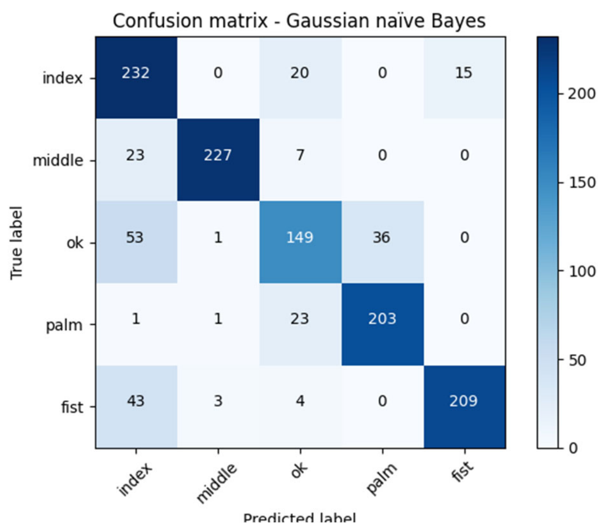


Figure 5.6: *Confusion matrix of Gaussian Naïve Bayes classifier model is the accuracy of participants 1-5 mixed with 75% train and 25% test*

Figure 5.6 shows the confusion matrix for Gaussian Naïve Bayes Classifier. The model has an average recognition rate of 81.6 %. We also performed the same test with the Decision tree classifier. As seen in Figure 5.7, the decision tree classifier results are improved compared to Gaussian Naïve Bayes Classifier. In fact, when compared to multiple different types of probabilistic classifiers such as SVM (~76 %), KNN (~67 %), and Naïve Bayes Classifiers (~81.6 %), the decision represents one of the best results with an average recognition rate of 91.1 %

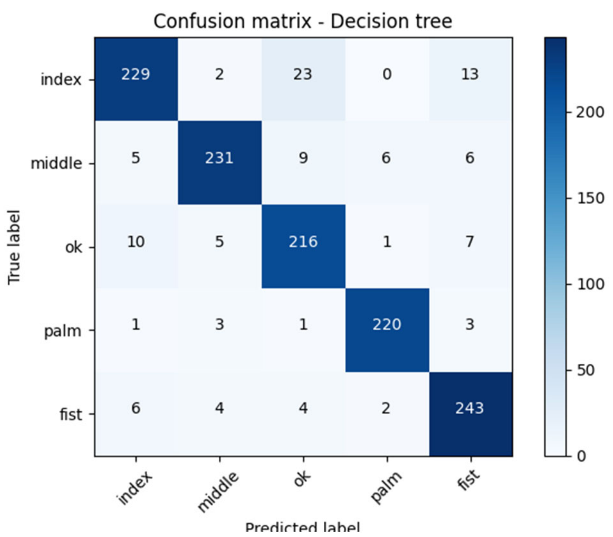


Figure 5.7: *Confusion matrix of Fine decision tree classifier, left is model accuracy of participants 1-5 mixed together with 75% train/validation and 25% test, right is participants 1-4 train/validate and participant 5 used for test.*

This high accuracy suggests that a simple Decision tree classifier can have high classification accuracy if the model uses data captured in the same setting for both training and testing. However, while the decision tree model shows a significant gain compared to Gaussian Naïve Bayes Classifier, the classification accuracy cannot compare to even simple neural networks, as will be seen later.

5.3.1 Independent Training and Testing

The next step was to test the probabilistic classifiers model with independent train/validation and test data. We will use the data captured from four subjects for training/validation and the data captured with the fifth participant to test. This is a more robust evaluation compared to the previous method where the data from all 5 participants were split into train and test sets.

5.3.1.1 Gaussian Naïve Bayes

Figure 5.8 shows the classification accuracy results of Gaussian Naïve Bayes with such a setup. The results of all the models are summarized in Table 3. It can be seen from Figure 5.8 that model accuracy has significant variations depending on which data is used for training/validation and what is used for testing. The lowest accuracy was only 63.3% with Train/validation participants 1, 2, 3, 4, and Test data participant 5. The highest model accuracy of 82.5 % was observed with participants 1, 3, 4, and 5 as Train/validation and testing with participant 2. This significant variation between models with different participants combinations shows the varying nature of capacitance sensors and the limited capability of Gaussian Naïve Bayes with an average accuracy of only 70.86 %.

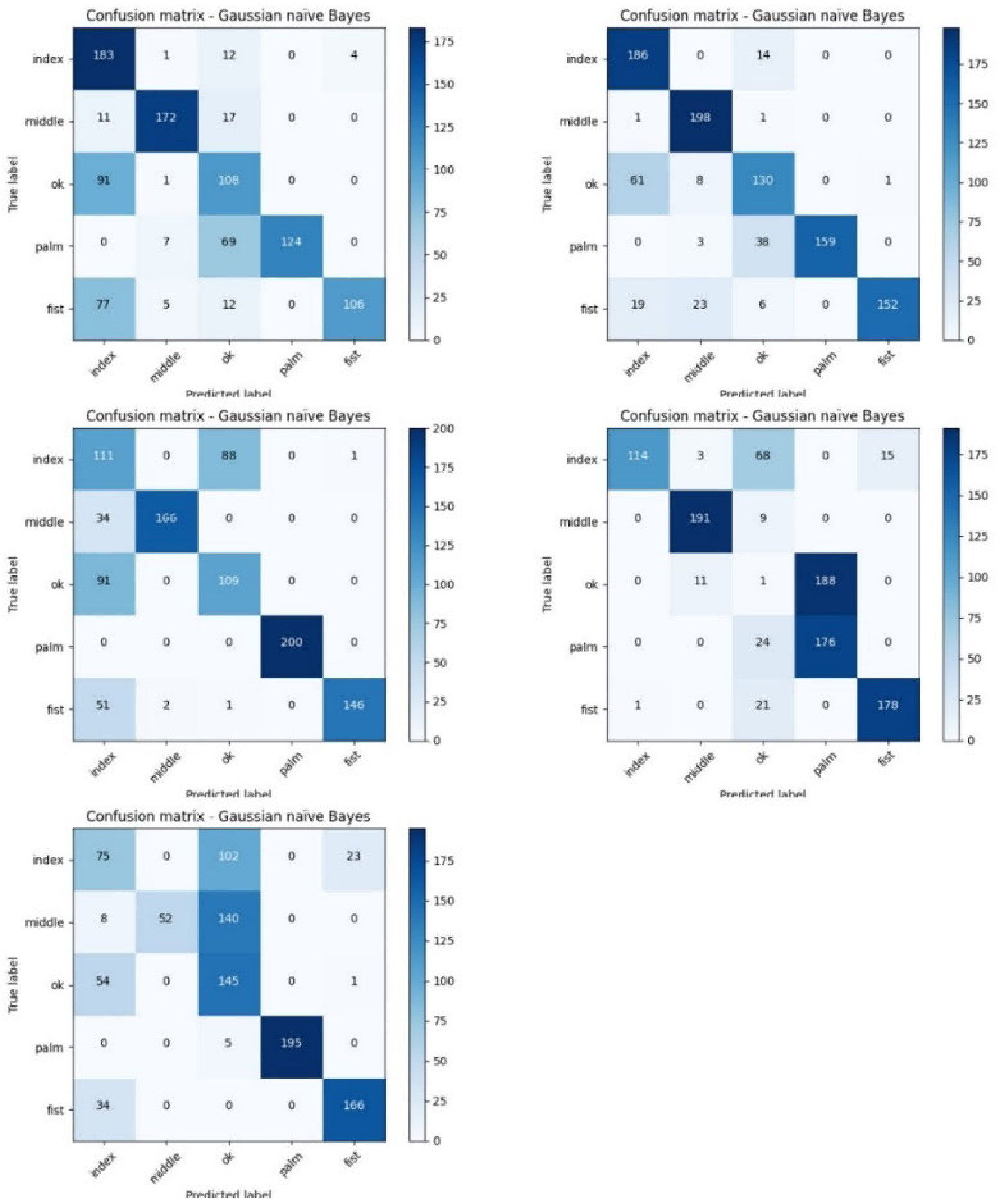


Figure 5.8: *All Gaussian Naïve Bayes combination results, top left: training/validation with participant 2,3,4,5 test with participant 1. Top right: training/validation with participants 1,3,4,5, test with participant 2. Middle left: training/validation with participant 1,2,4,5, test with participant 3. Middle right : training/validation with participant 1,2,3,5, test with participant 4. Bottom left: training/validation with participants 1,2,3,4, test with participant 5*

Table 3: *results of Gaussian Naïve Baye with different participants combination*

Result relative to Figure 5.8	Train/validation participants	Test data participant	Classification accuracy
Top left	2, 3, 4, 5	1	69.3 %
Top Right	1, 3, 4, 5	2	82.5 %
Middle Left	1, 2, 4, 5	3	73.2 %
Middle Right	1, 2, 3, 5	4	66.0 %
Bottom Left	1, 2, 3, 4	5	63.3 %
Average			70.86 %

5.3.1.2 Decision tree

Figure 5.9 shows the classification accuracy and results of the decision tree classifier. All results are summarized in Table 4. From Table 4 and Figure 5.9, it can be seen that the models have more minor variations than Gaussian Naïve Bayes models but still has some variations. This time the highest accuracy was observed with Train/validation participants 1, 2, 4, 5, and Test data participant 3 at 85.6 %. the average of the decision tree classifier is 77.86 % which shows a marked increase in 7.0 % accuracy compared to Gaussian Naïve Bayes models 70.86 % and thus is better suited to be used for our dataset. Yet as we will see, neural networks will decidedly improve upon these results.

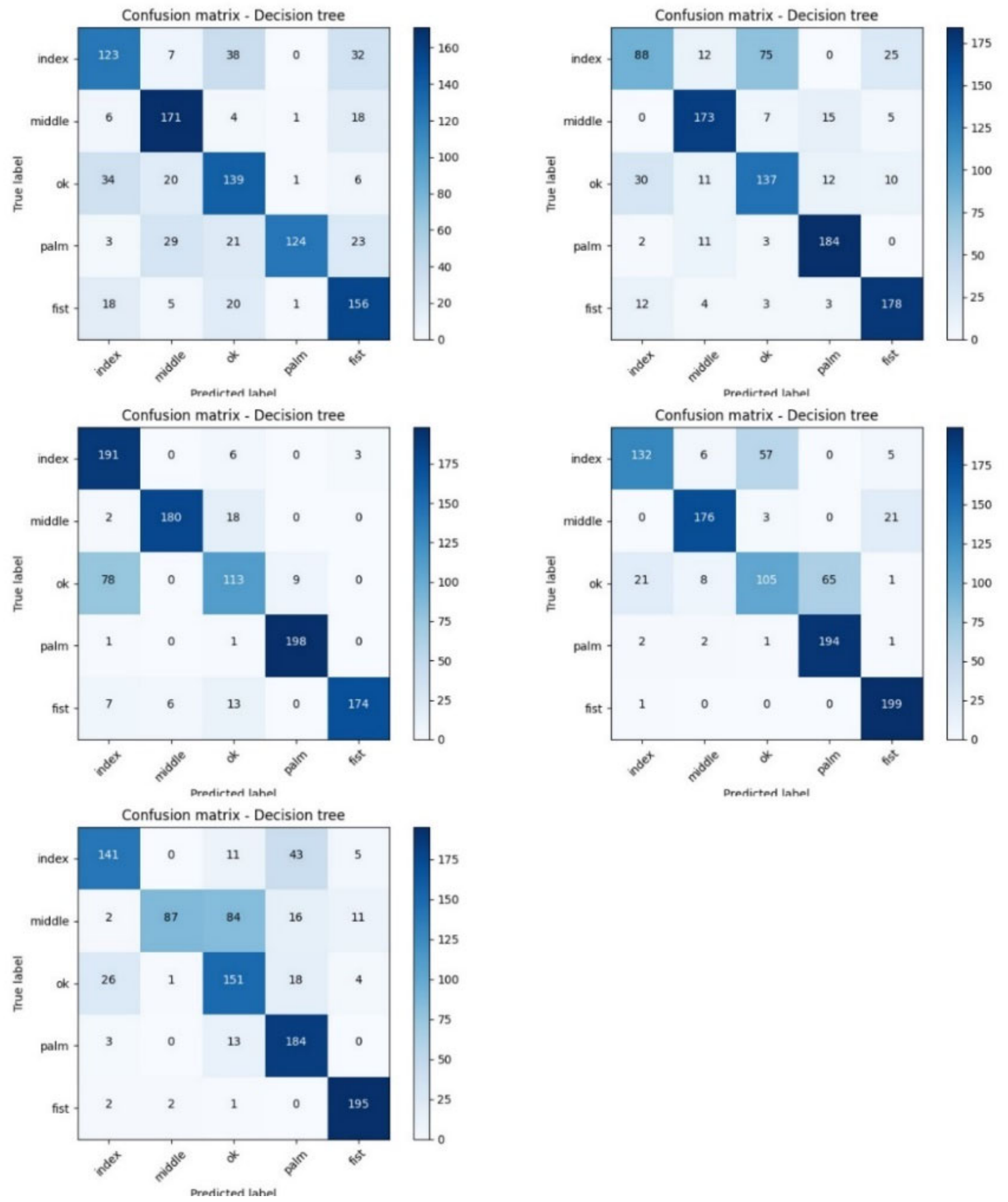


Figure 5.9: *All decision tree combination results, top left: training/validation with participant 2,3,4,5 test with participant 1. Top right: training/validation with participants 1,3,4,5, test with participant 2. Middle left: training/validation with participant 1,2,4,5, test with participant 3. Middle right : training/validation with participant 1,2,3,5, test with participant 4. Bottom left: training/validation with participants 1,2,3,4, test with participant 5.*

Table 4: *Results of decision tree with different participants combination*

Result relative to Figure 5.9	Train/validation participants	Test data participant	Classification accuracy
Top left	2, 3, 4, 5	1	71.3 %
Top Right	1, 3, 4, 5	2	76.0 %
Middle Left	1, 2, 4, 5	3	85.6 %
Middle Right	1, 2, 3, 5	4	80.6 %
Bottom Left	1, 2, 3, 4	5	75.8 %
Average			77.86 %

5.4 Multilayer Perceptron Results

5.4.1 Performance for Train/Test Split Containing all Subjects

The design of our MLP is shown in Figure 5.10. The design is a sequential two-layer MLP with two hidden layers using Rectified Linear Units (ReLU) [61]. It should be noted that we did not see a significant impact when using other activation functions such as tanh. Two hidden layers are chosen to allow sufficient depth to the network. 3 and 4 layers MLP designs were tested but did not show improvement compared to the 2 hidden layers MLP design while training time was significantly increased. The node setup is with 100 nodes for the first hidden layer and 50 for the second layer, the choice of 100 nodes for the first layer is set through an estimate of the input layer plus output layer and then fine-tuning. With around 60 nodes for the first hidden layer, underfitting of the model can be observed, and while going as high as 300 nodes do not seem to induce overfitting, it significantly increases the training time and does not increase model accuracy. For the second layer, 50 nodes are chosen as we did not see underfitting until 20 nodes, and no overfitting was observed at 100 nodes. Therefore 50 was selected as a safety margin between safety and training time.

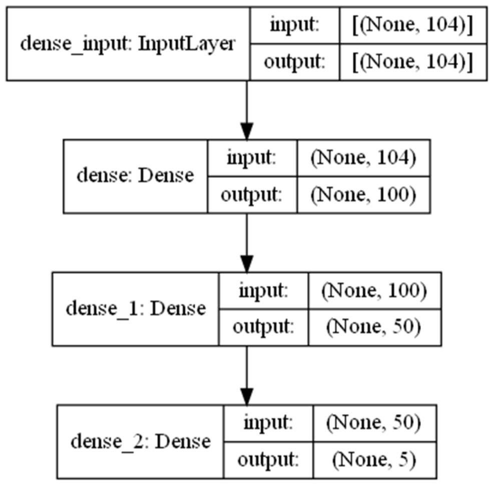


Figure 5.10: *MLP model used in our design*

No dropout or variable training rate is used for this first MLP design. The loss equation used is sparse Categorical Cross-entropy, and the optimizer used is SGD Gradient descent with momentum optimizer (SGD). Epoch is selected at 100 to detect overtraining, and batch size is chosen at 16.

To test this MLP design, we combined the data of all 5 participants' data frames and then randomly performed a 70/15/15 split—70 % for training, 15 % for validation and 15 % for testing. The purpose of this classification is to establish a baseline result compared to probabilistic classifiers to see if neural networks produce better results than probabilistic classifiers.

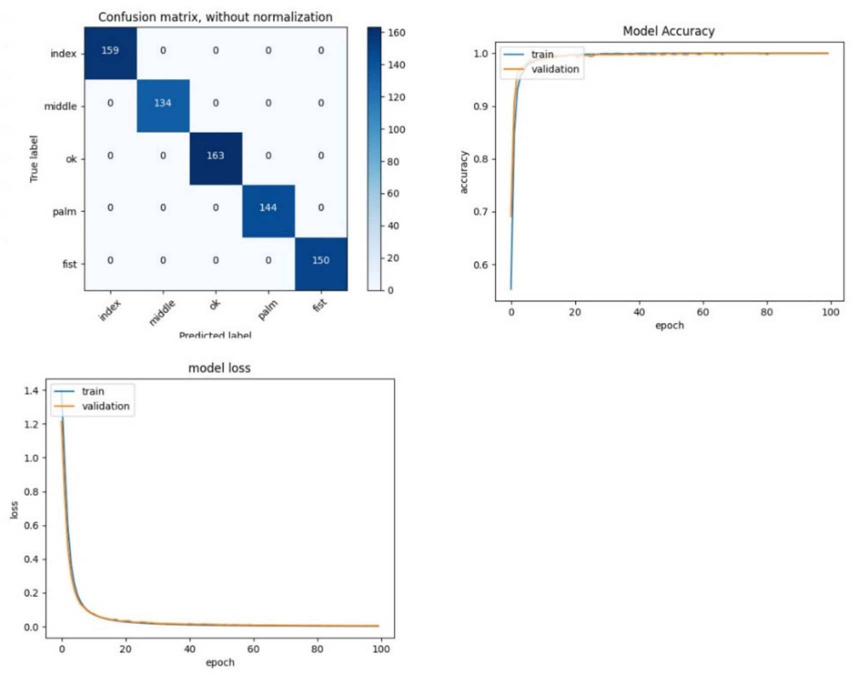


Figure 5.11: *MLP result of mixing dataset 3 participants 1-5 together*

Figure 5.11 shows the results when all the participants 1-5 from dataset three are mixed, and then a 70/15/15 split is performed. This result shows a 100 % classification rate with no overfitting or overtraining; the model loss also converges perfectly.

5.4.2 Independent Training and Testing Data

Naturally, the next logical step would be to use four sets of test participants' data as training/validation and 1 set of test participants' data as testing as it was done for simple probabilistic classifiers. Models containing multiple combinations are tested to show the highest and lowest accuracy.

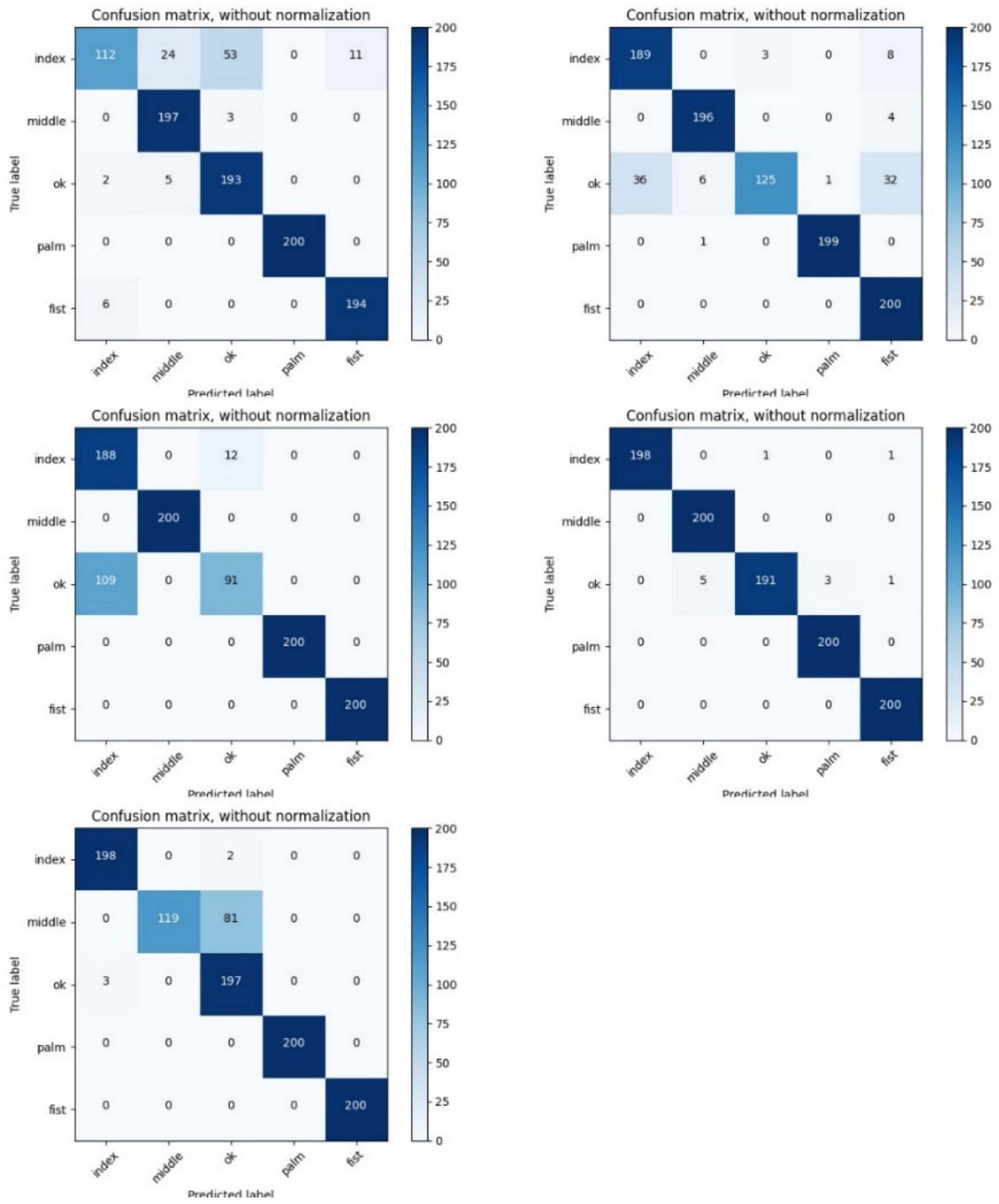


Figure 5.12: *All MLP combination results, top left: training/validation with participant 2,3,4,5 test with participant 1. Top right: training/validation with participant 1,3,4,5, test with participant 2. Middle left: training/validation with participant 1,2,4,5, test with participant 3. Middle right: training/validation with participant 1,2,3,5, test with participant 4. Bottom left: training/validation with participant 1,2,3,4, test with participant 5*

Figure 5.12 shows all possible combinations where 4 participants are used for training/validation and 1 for testing. The accuracy of all the models is summarized in Table 5. From the table, it can be seen that accuracy can range from 87.9 % to 98.9 % success rate with an average of 91.7 %. The epoch at which these accuracies is achieved is also quite spread, ranging from 21 to 62. From the confusion matrix, it can also be noted that the difference in combination of training/validation and testing participants produces different wrong classifications; all five models have no misclassification of the fist and palm gesture and seem to have a significant issue in classifying the Ok gesture and Index gesture. The top right and left middle models have particular problems with this ok gesture where large amounts are incorrectly classified as the index gesture. This difference in accuracy, epoch, and classification errors observed in different models gives us confidence that there are still variations in the data due to hand size variation or environmental variables.

Table 5: *Results of MLP with different participants combinations*

Result relative to Figure 5.12	Train/validation participants	Test data participant	Successful classification percentage	F-score
Top left	2, 3, 4, 5	1	89.6 %	89.0 %
Top Right	1, 3, 4, 5	2	90.9 %	90.0 %
Middle Left	1, 2, 4, 5	3	87.9 %	87.0 %
Middle Right	1, 2, 3, 5	4	98.9 %	99.0 %
Bottom Left	1, 2, 3, 4	5	91.4 %	91.0 %
Average			91.7 %	91.2 %

5.4.3 Impact of Dropouts

Dropout works by probabilistically removing, or “dropping out,” inputs to a layer, which may be input variables in the data sample or activations from a previous layer. It has the effect of simulating a large number of different input scenarios and makes nodes in the network generally more robust to the inputs.

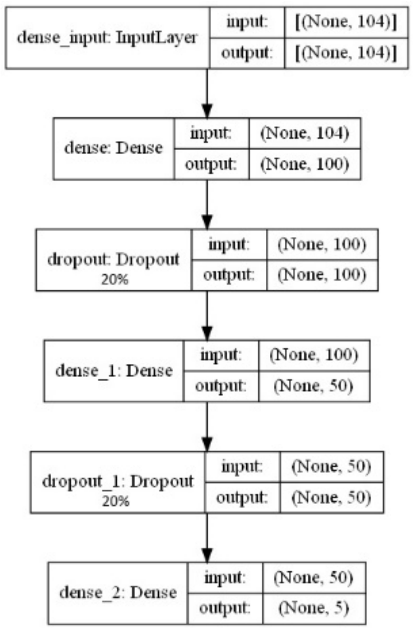


Figure 5.13: *MLP model setup with dropout*

To test if dropout will create more linear models and higher classification accuracy, we will test it with a similar data structure to section 5.4.2 with four sets of test participants' data as training/validation and 1 set of test participants' data as testing. Figure 5.13 the MLP set up with dropout added. The dropout occurs after the two hidden layers, where 20% are dropped at each point. This dropout is very aggressive in the hopes to identify any points of interest or that if dropout has a significant effect on the model outcome.

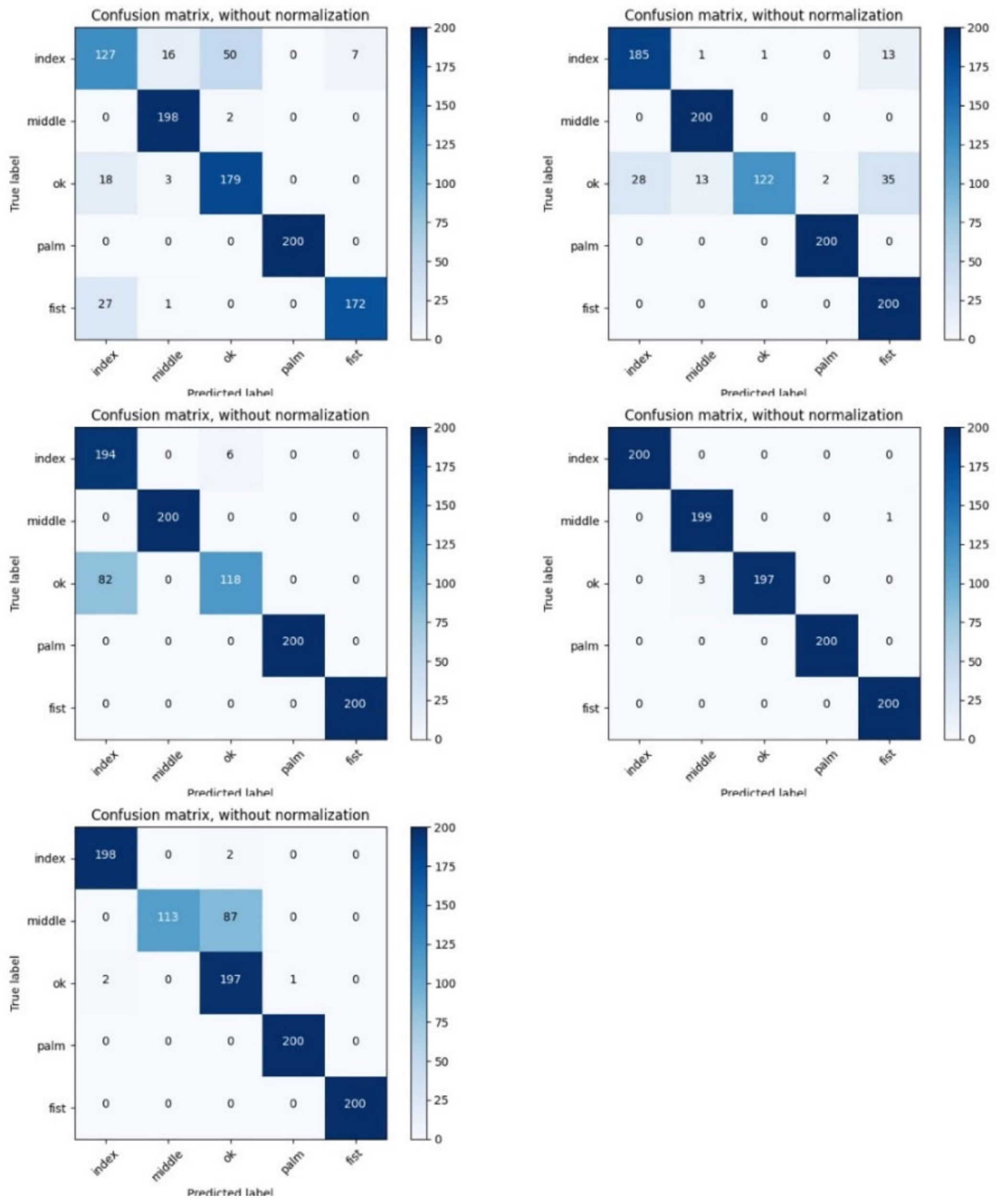


Figure 5.14: *MLP with dropout combination results, top left: training/validation with participant 2,3,4,5 test with participant 1. Top right: training/validation with participant 1,3,4,5, test with participant 2. Middle left: training/validation with participant 1,2,4,5, test with participant 3. Middle right : training/validation with participant 1,2,3,5, test with participant 4. Bottom left: training/validation with participant 1,2,3,4, test with participant 5*

Figure 5.14 shows data results of MLP data with dropout. All models show a modest gain in accuracy over non-dropout models in most cases. Results of all the models with dropout are summarized in Table 6. The highest increase over non-drop out was observed with Train/validation participants 1, 2, 4, 5, and Test data participant 3, with an increase at 3.3 %.

The overall accuracy is 92.34 %, Which represents an 0.6 % average increase over non-dropout MLP (91.74 %).

Table 6: *Results of MLP with dropouts with different participants combinations*

Result relative to Figure 5.14	Train/validation participants	Test data participant	Successful MLP classification percentage	F-Score
Top left	2, 3, 4, 5	1	89.4 %	88.0 %
Top Right	1, 3, 4, 5	2	90.7 %	90.0 %
Middle Left	1, 2, 4, 5	3	91.2 %	91.0 %
Middle Right	1, 2, 3, 5	4	99.6 %	98.0 %
Bottom Left	1, 2, 3, 4	5	90.8 %	93.0 %
Average			92.34%	92.0 %

The middle right model with Train/validation participants 1, 2, 3, 5, and participant 1 as test data gives us the most accurate classification for MLP. We also tested dropouts added to unprocessed data (before dense_input in Figure 5.13), where 20.0 % of incoming data is readily removed before even the first layer. This removal, as expected, provides model accuracies that are worse than even MLP without any dropout.

5.5 Convolutional Neural Network Results

The convolutional neural network was also tested to see if it yielded improved results than MLP. As CNN usually shows improvements over MLP when it comes to image-based classifying, one would expect improved results in our case as we treat our data frames as low-resolution grayscale images. To test how CNN would perform, we would first have to reorganize the data back to the correct position as per it was on the sensor pads. We need to do this to simplify data processing for MLP; the data that comes from the STM32F7 is the form of a single row for a single data frame. CNN requires the 2D array of the original image to be effective and therefore, we must reorganize the data back to its 2D format. Therefore, the first step is to turn the single row into 2D array data where each cell represents a single capacitor sensor on the sensor board.

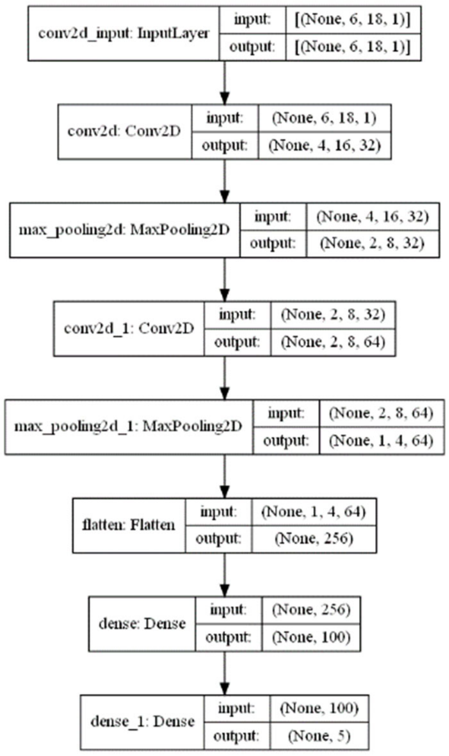


Figure 5.15: *CNN model setup*

Figure 5.15 shows the setup used for the CNN model. As our model does not have data from different channels such as that of an RGB image, thus a single 6 by 18 by 1 matrix is sufficient to describe our data frame, 6 by 18 represents our 6 by 18 sensor board (2 blank pads), and 1 represents our grayscale nature. One hidden layer of 100 is chosen for simplicity.

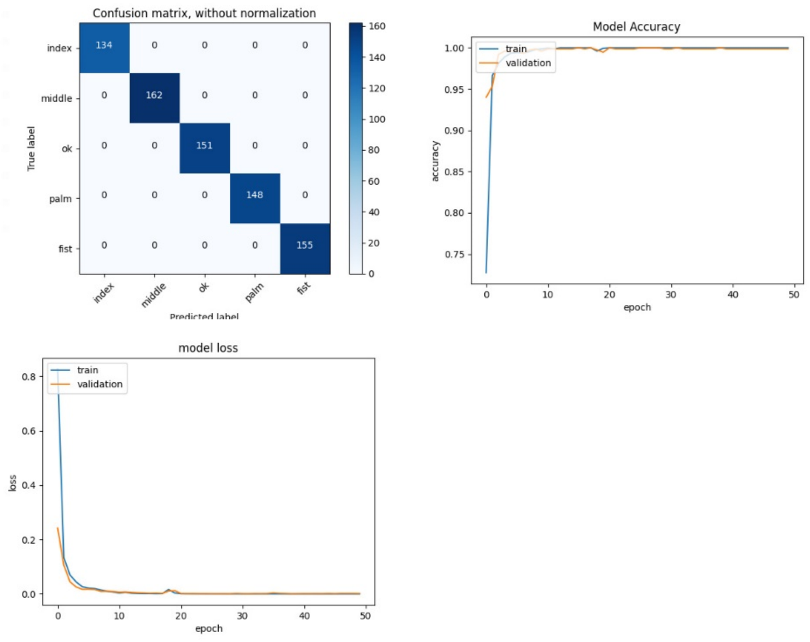


Figure 5.16: *CNN results of the result of mixing dataset 3 participants 1-5 together*

Figure 5.16 shows the results of the CNN model used, as it shows that when the model is trained, validated, and tested with 70/15/15 split with mixed data from the participants 1-5, no overfitting or overtraining is observed, and the model accuracy is 100.0 %. This result shows that CNN can be a capable form of neural network design in our use case. But whether CNN is an improvement or degrades over MLP models remains to be seen as MLP models also achieved a 100.0 % classification accuracy when mixing data from participants 1-5.

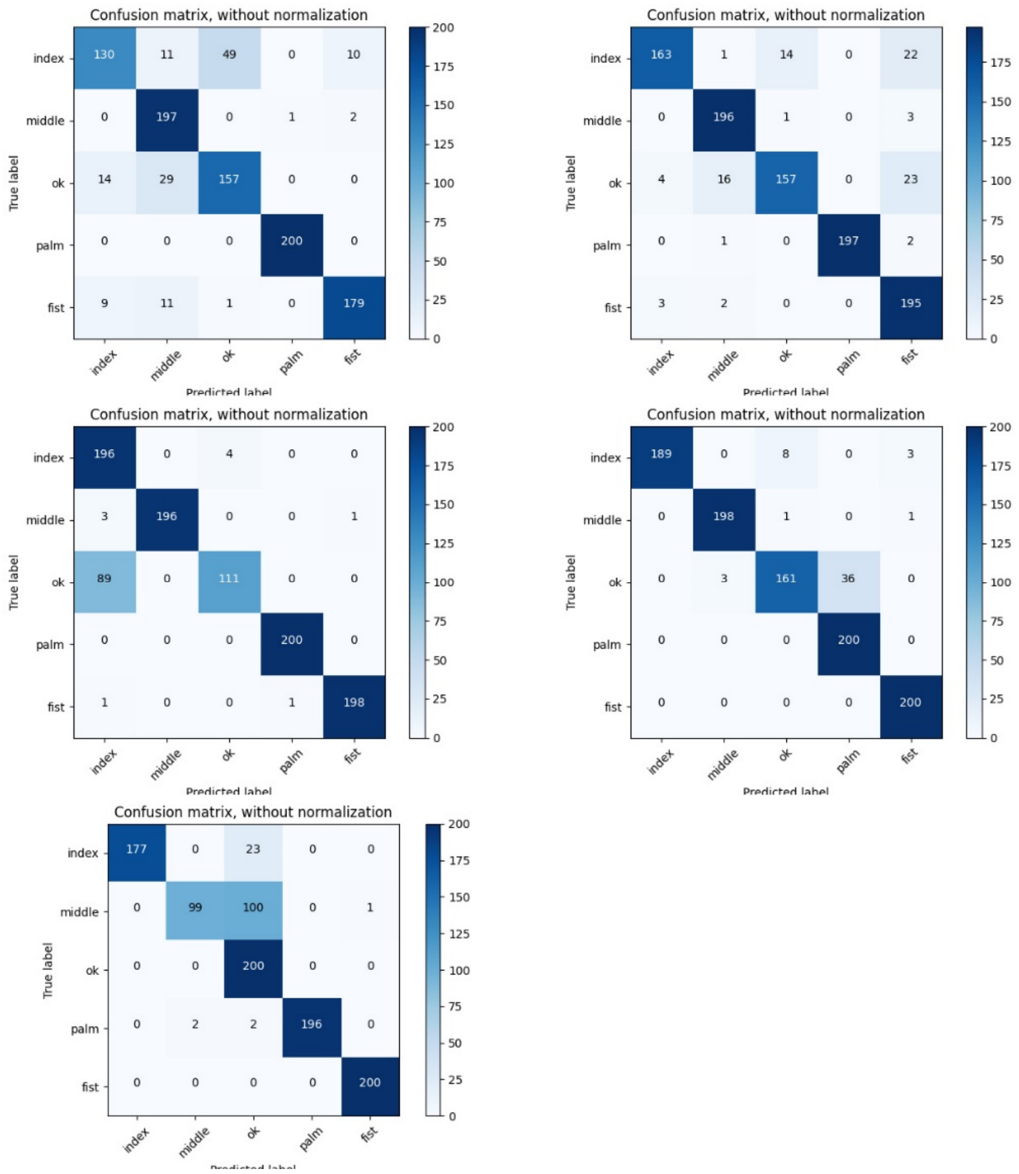


Figure 5.17: *CNN combination results, top left: training/validation with participant 2,3,4,5 test with participant 1. Top right: training/validation with participant 1,3,4,5, test with participant 2. Middle left: training/validation with participant 1,2,4,5, test with participant 3. Middle right: training/validation with participant 1,2,3,5, test with participant 4. Bottom left: training/validation with participant 1,2,3,4, test with participant 5*

Figure 5.17 shows all possible combinations of CNN where 4 participants are used for training/validation and 1 for testing. The accuracy of all the models is summarized in Table 7. From the table, it can be seen that accuracy can range from 86.3 % to 94.8 % success rate, the average being 90.40 %. This result shows that this result is comparable but overall worse than MLP results. The average for MLP is 91.74 % without dropout and 92.34 % with dropout, which represents a 1.34 % and 1.94 % increase respectively. Reasons for this can be due to the grayscale nature of our data, and therefore not sufficient data is present for CNN to perform better than MLP in our models or that CNN's advantages of spatially invariant were not utilized; however, the research to prove this theory was not in the scope of this thesis. From this data, it was decided that MLP is sufficient for our use case, and therefore much of further experiments will be based on MLP instead of dividing resources between MLP and CNN.

Table 7: *results of CNN with different participants combinations*

Result relative to Figure 5.14	Train/validation participants	Test data participant	Successful CNN classification percentage	F-Score
Top left	2, 3, 4, 5	1	86.3 %	86.0 %
Top Right	1, 3, 4, 5	2	90.4 %	91.0 %
Middle Left	1, 2, 4, 5	3	90.1 %	90.0 %
Middle Right	1, 2, 3, 5	4	94.8 %	95.0 %
Bottom Left	1, 2, 3, 4	5	87.2 %	87.0 %
Average			90.40 %	89.8%

5.6 Conclusion

All results of all test models are summarized in Table 8. From the overview, it can be seen that probabilistic classifiers perform worse than neural networks as a whole. The simple decision tree classifier can achieve 91.12 % classification accuracy when presented with completely mixed data, while Naïve Bayes can only achieve 81.6 %. However, when neural networks are presented with similar data, all models are capable of achieving 100.0 % accuracy.

Table 8: *All model results summary*

Train/validation participants	Test participant	Naïve Baye	Simple decision tree	MLP without dropout (F-score)	MLP with dropout (F-score)	CNN (F-score)
75 % data from all 5 users	15 % data from all 5 users	81.6 %	91.1 %	100.0 % (100.0 %)	100.0 % (100.0 %)	100.0 % (100.0 %)
2, 3, 4, 5	1	69.3 %	71.3 %	89.6 % (89 %)	89.4 % (88.0 %)	86.3 % (86.0 %)
1, 3, 4, 5	2	82.5 %	76.0 %	90.9 % (90 %)	90.7 % (90.0 %)	90.4 % (91.0 %)
1, 2, 4, 5	3	73.2 %	85.6 %	87.9 % (87 %)	91.2 % (91.0 %)	90.1 % (90.0 %)
1, 2, 3, 5	4	66.0 %	80.6 %	98.9 % (99 %)	99.6 % (98.0 %)	94.8 % (95.0 %)
1, 2, 3, 4	5	63.3 %	75.8 %	91.4 % (91 %)	90.8 % (93.0 %)	87.2 % (87.0 %)
Average (training/validation with any 4 subjects and test on the remaining subject)		70.9 %	77.86 %	91.74 % (91.2 %)	92.34 % (92 %)	90.40 % (89.8 %)

When comparing between MLP and CNN, it can be seen that CNN has overall less accuracy than MLP with a drop of 1.34 % as an average. This loss in accuracy can be maybe attributed to the single-dimensional nature of the capacitance data, where CNN's key advantages of spatially invariant and its attempt to reduce dimensions of images may not be valuable for processing our dataset. When adding dropout to MLP, there are mostly accuracy improvements to models over the non-dropout MLP models. This increase is not universal as in the case of Train/validation participants 1, 2, 3, 4 and Test participant 5, we see a drop in accuracy. However, this accuracy difference can simply be due to variations due to the random weight assignment in the first iteration. Overall, Dropout has benefits compared to non-dropout-based models with an average increase of 0.6 % in overall accuracy.

CHAPTER 6: CONCLUSION AND FUTURE WORK

The purpose of this study was to evaluate the feasibility of capacitance sensors as a form of gesture recognition. The developed capacitive sensor system with a trained MLP classifier is capable of achieving an accuracy of 92.34 %. While the accuracy of the system outlined in this work does not match the levels of video cameras, the hardware involved in capacitance sensing is also much less complex than video based solutions. Further advantages of capacitance-based sensing are also evident with the low computational requirement, and inherent high privacy.

The limiting factors of the hardware system proposed in this work are the limited number of sensors and the large sensor pads, which limited resolution. With a smaller and more sensitive sensor design, we can hope to capture even higher resolution gesture data images. Isolation of the FDC1004 routing can be further improved upon with a higher layer count PCB and better dielectric. In theory, an isolated power supply for each FDC1004 can also significantly decrease the coupling between different FDC1004 as each excitation signal generated will not have a common ground. However, the cost of such a design will also increase. More Complex custom capacitance sensing methods can also be implemented, such as a charge balancer circuit or frequency measurement-based circuit. These custom designs allow much more extensive customization and freedom of design compared to off-the-shelf IC chips employed in this experiment.

Another limiting factor in the hardware is the way sensor measurements are captured. Due to the noise crosstalk of FDC1004, we can only trigger sensors one at a time. With further improvement, it may be possible to trigger all sensors at once. This simultaneous triggering of sensors would massively increase the speed at which hand gestures are captured. Furthermore, if the capture or sampling rate is fast enough, non-static gesture detection would also be possible.

An FPGA based design would significantly increase the entire system's processing speed, data bandwidth and reduce the design cost from a component count perspective. The current hardware uses a custom communication protocol to accommodate for the massive number of MCU used. By using an FPGA in place of all the MCUs we can hope to replace communication protocol with industry standard methods and thus increase the system's integrity significantly.

Other modes of capacitor modes can also be explored. As this work only explored loading mode, a deeper look into shunt mode can also be interesting as shunt mode offers inherently more coupling immunity than loading mode at the cost of increased sensor pad footprint. Readily available IC chips from the likes of Analog Devices can allow rapid prototyping of shunt mode designs.

From the software side, improvements can be made in how data is processed. Currently, raw values captured by the sensors are passed straight to classifiers for classification. One can apply feature detection or bounding box-based approaches before data is fed to the classifier. By using a region of interest, we can further reduce the effects of the background noise that capacitance sensors capture, furthermore incoming data can also be applied with a non-linear filter to accentuate features while reducing the effects of noise.

REFERENCES

1. Dix, A. (2010). Human–computer interaction: A stable discipline, a nascent science, and the growth of the long tail. *Interacting with Computers, 22*(1), 13-27. https://doi.org/10.1016/j.intcom.2009.11.007

2. Ibraheem, N. A., & Khan, R. Z. (2012). Survey on various gesture recognition technologies and techniques. *International journal of computer applications, 50*(7).

3. Liu, H., & Wang, L. (2018). Gesture recognition for human-robot collaboration: A review. *International Journal of Industrial Ergonomics, 68*, 355-367. https://doi.org/10.1016/j.ergon.2017.02.004

4. Google Project Soli. Retrieved 11/07 from https://atap.google.com/soli/

5. Suarez, J., & Murphy, R. R. (2012). *Hand gesture recognition with depth images: A review* https://dx.doi.org/10.1109/roman.2012.6343787

6. Garg, P., Aggarwal, N., & Sofat, S. (2009). Vision based hand gesture recognition. *World academy of science, engineering and technology, 49*(1), 972-977. ; Murthy, G., & Jadon, R. (2009). A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management, 2*(2), 405-410.

7. Starner, T. E. (1995). *Visual Recognition of American Sign Language Using Hidden Markov Models*. ; Starner, T., Weaver, J., & Pentland, A. (1998). Real-time American sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(12), 1371-1375. https://doi.org/10.1109/34.735811

8. Howe, N., Leventon, M. E., & Freeman, W. T. (2000). Bayesian reconstruction of 3d human motion from single-camera video. *Advances in neural information processing systems, 12*, 820.

9. Mitra, S., & Acharya, T. (2007). Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), 37*(3), 311-324. https://doi.org/10.1109/tsmcc.2007.893280

10. Xianghua, L., Jun-Ho, A., Jin-Hong, M., & Kwang-Seok, H. (2011). *Hand gesture recognition by stereo camera using the thinning method* https://dx.doi.org/10.1109/icmt.2011.6001670

11. Soutschek, S., Penne, J., Hornegger, J., & Kornhuber, J. (2008). *3-D gesture-based scene navigation in medical imaging applications using Time-of-Flight cameras* https://dx.doi.org/10.1109/cvprw.2008.4563162

12. Dipietro, L., Sabatini, A. M., & Dario, P. (2008). A Survey of Glove-Based Systems and Their Applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38*(4), 461-482. https://doi.org/10.1109/tsmcc.2008.923862

13. Condell, J., Curran, K., Quigley, T., Gardiner, P., McNeill, M., Winder, J., Xie, E., & Qi, Z. (2010). Measuring Finger Movement in Arthritic Patients Using Wearable Glove Technology. In *Wearable and Autonomous Biomedical Devices and Systems for Smart Environment* (pp. 391-406). Springer.

14. Oshiumi, K. (2004). INDUCTCODER Principle and Applications. *Journal of The Japan Institute of Marine Engineering, 39*(11), 752-758. https://doi.org/10.5988/jime.39.752

15. Kuroda, T., Tabata, Y., Goto, A., Ikuta, H., & Murakami, M. (2004). *Consumer price data-glove for sign language recognition* Proc. ICDVRAT,

16. Caeiro-Rodríguez, M., Otero-González, I., Mikic-Fonte, F. A., & Llamas-Nistal, M. (2021). A Systematic Review of Commercial Smart Gloves: Current Status and Applications. *Sensors, 21*(8), 2667. https://doi.org/10.3390/s21082667

17. Zhang, Y., & Harrison, C. (2015). *Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition* Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, Charlotte, NC, USA. https://doi.org/10.1145/2807442.2807480

18. Way, D., & Paradiso, J. (2014). *A usability user study concerning free-hand microgesture and wrist-worn sensors* 2014 11th International Conference on Wearable and Implantable Body Sensor Networks,

19. Grosse-Puppendahl, T., Holz, C., Cohn, G., Wimmer, R., Bechtold, O., Hodges, S., Reynolds, M. S., & Smith, J. R. (2017). *Finding Common Ground: A Survey of Capacitive Sensing in Human-Computer Interaction* https://dx.doi.org/10.1145/3025453.3025808

20. Grosse-Puppendahl, T., Berghoefer, Y., Braun, A., Wimmer, R., & Kuijper, A. (2013). *OpenCapSense: A rapid prototyping toolkit for pervasive interaction using capacitive sensing* https://dx.doi.org/10.1109/PerCom.2013.6526726

21. Texas Instruements (2014). FDC1004 4-Channel Capacitance-to-Digital Converter. Retrieved May 2020 from https://www.ti.com/product/FDC1004

22. Buller, W., & Wilson, B. (2006). Measurement and Modeling Mutual Capacitance of Electrical Wiring and Humans. *IEEE Transactions on Instrumentation and Measurement, 55*(5), 1519-1522. https://doi.org/10.1109/tim.2006.880293

23. Kim, S., Kawahara, Y., Georgiadis, A., Collado, A., & Tentzeris, M. M. (2013). *Low-cost inkjet-printed fully passive RFID tags using metamaterial-inspired antennas for capacitive sensing applications* https://dx.doi.org/10.1109/MWSYM.2013.6697644; Li, H., Brockmeyer, E., Carter, E. J., Fromm, J., Hudson, S. E., Patel, S. N., & Sample, A. (2016). *PaperID: A Technique for Drawing Functional Battery-Free Wireless Interfaces on Paper* https://dx.doi.org/10.1145/2858036.2858249; Singh, G., Nelson, A., Robucci, R., Patel, C., & Banerjee, N. (2015). *Inviz: Low-power personalized gesture recognition using wearable textile capacitive sensor arrays* https://dx.doi.org/10.1109/percom.2015.7146529

24. Smith, J. R. (1996). Field mice: Extracting hand geometry from electric field measurements. *IBM Systems Journal, 35*(3.4), 587-608. https://doi.org/10.1147/sj.353.0587

25. Glinsky, A. (2000). *Theremin : ether music and espionage*. University of Illinois Press. http://books.google.de/books?id=6DHlQJcMpBQC

26. Mizumoto, T., Tsujino, H., Takahashi, T., Ogata, T., & Okuno, H. G. (2009). *Thereminist robot: Development of a robot theremin player with feedforward and feedback arm control based on a Theremin's pitch model* https://dx.doi.org/10.1109/IROS.2009.5354473

27. Zimmerman, T. G., Smith, J. R., Paradiso, J. A., Allport, D., & Gershenfeld, N. (1995). *Applying electric field sensing to human-computer interfaces* https://dx.doi.org/10.1145/223904.223940

28. Tang, X., & Mandal, S. (2019). Indoor Occupancy Awareness and Localization Using Passive Electric Field Sensing. *IEEE Transactions on Instrumentation and Measurement, 68*(11), 4535-4549. https://doi.org/10.1109/tim.2018.2890319

29. Smith, J. R. (1999). *Electric field imaging* Massachusetts Institute of Technology]. https://dspace.mit.edu/handle/1721.1/29144

30. Reverter, F., Li, X., & Meijer, G. C. M. (2006, 2006/09/28). Stability and accuracy of active shielding for grounded capacitive sensors. *Measurement Science and Technology, 17*(11), 2884-2890. https://doi.org/10.1088/0957-0233/17/11/004

31. Grosse-Puppendahl, T., Beck, S., Wilbers, D., Zeiß, S., Von Wilmsdorff, J., & Kuijper, A. (2014). Ambient Gesture-Recognizing Surfaces with Visual Feedback. In (pp. 97-108). Springer International Publishing. https://doi.org/10.1007/978-3-319-07788-8_10

32. Smith, J. R. (1995). *Toward electric field tomography* Massachusetts Institute of Technology]. https://dspace.mit.edu/bitstream/handle/1721.1/62334/34312119-MIT.pdf?sequence=2

33. Valtonen, M., Maentausta, J., & Vanhala, J. (2009). *TileTrack: Capacitive human tracking using floor tiles* https://dx.doi.org/10.1109/PERCOM.2009.4912749

34. Rekimoto, J. (2002). *SmartSkin* https://dx.doi.org/10.1145/503376.503397

35. Habib, I., Berggren, N., Rehn, E., Josefsson, G., Kunz, A., & Fjeld, M. (2009). DGTS: Integrated Typing and Pointing. In (pp. 232-235). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-03658-3_30

36. Akhtar, H., & Kakarala, R. (2014). A Methodology for Evaluating Accuracy of Capacitive Touch Sensing Grid Patterns. *Journal of Display Technology, 10*(8), 672-682. https://doi.org/10.1109/jdt.2014.2312975

37. Dietz, P., & Leigh, D. (2001). *DiamondTouch* https://dx.doi.org/10.1145/502348.502389

38. Mayton, B., Legrand, L., & Smith, J. R. (2010). *An Electric Field Pretouch system for grasping and co-manipulation* https://dx.doi.org/10.1109/ROBOT.2010.5509658; Jonassen, N. (1998). *Human body capacitance: static or dynamic concept? [ESD]* https://dx.doi.org/10.1109/EOSESD.1998.737028

39. Wimmer, R., Kranz, M., Boring, S., & Schmidt, A. (2007). *A Capacitive Sensing Toolkit for Pervasive Activity Detection and Recognition* https://dx.doi.org/10.1109/PERCOM.2007.1

40. Valtonen, M., & Vanhala, J. (2009). *Human tracking using electric fields* https://dx.doi.org/10.1109/PERCOM.2009.4912795

41. Microchip (2020). ATmega48A/PA/88A/PA/168A/PA/328/P. Retrieved June 2021 from https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf

42. STMicroelectronics (2020). STM32G031x4/x6/x8. Retrieved July 2020 from https://www.st.com/resource/en/datasheet/stm32g031j6.pdf

43. Samtec (2021). ERF8-060-05.0-L-DV-K-TR. Retrieved 2021 from https://www.samtec.com/products/erf8-060-05.0-l-dv-k-tr

44. Analog Devices (2020). LTC3603 2.5A, 15V Monolithic Synchronous Step-Down Regulator. Retrieved September 2020 from https://www.analog.com/media/en/technical-documentation/data-sheets/3603fc.pdf

45. Devices, A. (2022). LTpowerCAD and LTpowerPlanner. Retrieved 2/4/22 from https://www.analog.com/en/design-center/ltpowercad.html

46. Analog Devices (2019). ADP7158, 2A Ultralow Noise, High PSRR, RF Linear Regulator. Retrieved June 2021 from https://www.analog.com/media/en/technical-documentation/data-sheets/ADP7158.pdf

47. STMicroelectronics (2021). STM32H7 Nucleo-144 boards (MB1363). Retrieved July 2021 from https://www.st.com/resource/en/user_manual/dm00499171-stm32h7-nucleo144-boards-mb1363-stmicroelectronics.pdf

48. On Semiconductor (2013). NCP59150, NCV59150 Series. Retrieved June 2021 from https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwidrZu2yq_xAhUVVpQKHUipDq8QFjAAegQIBBAD&url=https%3A%2F%2Fwww.onsemi.com%2Fpdf%2Fdatasheet%2Fncp59150-d.pdf&usg=AOvVaw1B-fPh9Ht71my2HpsoXkZe

49. Texas Instruements (2015). CapacitiveSensing:Ins and Outsof ActiveShielding. Retrieved June 2021 from https://www.ti.com/lit/pdf/snoa926

50. Tzanos, G., Kachris, C., & Soudris, D. *Hardware Acceleration on Gaussian Naive Bayes Machine Learning Algorithm* https://dx.doi.org/10.1109/mocast.2019.8741875

51. Swain, P. H., & Hauska, H. (1977). The decision tree classifier: Design and potential. *IEEE Transactions on Geoscience Electronics, 15*(3), 142-147. https://doi.org/10.1109/tge.1977.6498972

52. Suk, H.-I. (2017). An Introduction to Neural Networks and Deep Learning. In (pp. 3-24). Elsevier. https://doi.org/10.1016/b978-0-12-810408-8.00002-x

53. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Let a biogeography-based optimizer train your Multi-Layer Perceptron. *Information Sciences, 269*, 188-209. https://doi.org/10.1016/j.ins.2014.01.038

54. Mathur, P. (2016). *A Simple Multilayer Perceptron with TensorFlow*. Retrieved 5/11 from https://medium.com/pankajmathur/a-simple-multilayer-perceptron-with-tensorflow-3effe7bf3466

55. *Basics of Multilayer Perceptron – A Simple Explanation of Multilayer Perceptron*. (2018). https://kindsonthegenius.com/blog/basics-of-multilayer-perceptron-a-simple-explanation-of-multilayer-perceptron/

56. Saha, S. (2018). *A Comprehensive Guide to Convolutional Neural Networks* Retrieved 5/11 from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

57. Education, I. C. (2020). *Convolutional Neural Networks*. Retrieved 5/11 from https://www.ibm.com/cloud/learn/convolutional-neural-networks

58. Workalemahu, T. (2008). Singular value decomposition in image noise filtering and reconstruction.

59. Schanze, T. (2017). Removing noise in biomedical signal recordings by singular value decomposition. https://www.degruyter.com/document/doi/10.1515/cdbme-2017-0052/pdf

60. Epps, B. P., & Krivitzky, E. M. (2019). Singular value decomposition of noisy data: noise filtering. *Experiments in Fluids, 60*(8). https://doi.org/10.1007/s00348-019-2768-4

61. Zhao, G., Zhang, Z., Guan, H., Tang, P., & Wang, J. *Rethinking ReLU to Train Better CNNs* https://dx.doi.org/10.1109/icpr.2018.8545612