Queen Mary
University of London

School of Electronic Engineering and Computer Science

Development of Wireless Tactile Sensing Modules For Internet of Things Applications

MSc Internet of Things (Data)

Author: Azraful Awal Hamim

Student ID: 130367349

Supervisor: Dr. Lorenzo Jamone

Date: 21st August, 2019

ACKNOWLEDGEMENTS

I would like to acknowledge my sincerest gratitude towards Dr. Lorenzo Jamone for granting me the opportunity to work on this project. I would like to thank him for his incredible support and guidance throughout this project.

I would like to thank all my professors, mentors and tutors at Queen Mary University of London, without whom this project may not have been possible. I would also like to thank the electronics laboratory manager, Kok Ho Huen, for his kind assistance in acquiring the components for this project. I would like to especially thank Pedro Ribeiro at INESC MN for assembling the sensor PCB and thoroughly guiding me to use it. I want to also thank research students, Joshua Brown and Wanlin Li, for helping me with the 3D printing processes.

Furthermore, I would like to thank my family, fiancé and friends, for always being there for me with their undying love and positive motivation.

ABSTRACT

Tactile sensing is an active research area because of its evident usefulness in the rapidly evolving domain of the Internet of Things. Enabling a wireless sensing capability of the tactile sensor, as implemented in this project, reduces the requirements of bulky wires to operate it. There are a wide range of applications in several industries for such a wireless tactile sensor. The sensor is low-cost, small in size and has a soft shape. Therefore, it can successfully integrate in remote applications. This Hall-effect sensor can measure 3-axis normal and shear forces, with high sensitivity. The soft sensor body was 3D printed using a soft elastic material with high elongation. The wireless data transfer is achieved through the integrated Wi-Fi technology of the ESP32 microcontroller, which is powered by a rechargeable battery. This wireless tactile sensing module uses minimal electronic components, and is presented in a 3D printed housing device. The wireless data is visualised using a Wi-Fi enabled smartphone. Unlike previous wireless tactile sensing solutions, this project presents a miniature and low-cost magnetic sensor with soft skin that can measure forces in three axes, and also gather data in real-time wirelessly.

Table of Contents

List of Figures & Tables

I. INTRODUCTION

Tactile sensing gathers data from physical interactions between a hardware device and its external environment. Cutaneous touch in biological systems detect stimuli from pain, temperature and mechanical activation, which consequently inspires the development of mechanisms for tactile sensors in artificial systems. We use tactile sensors on a daily basis in devices such as touchscreen smartphones, but they can also be found in complex systems such as prosthetics, home automation and robotics.

A. Background

Tactile perception can act as human skin for devices or robots to explore its external environment by the sense of touch. Dahiya et al. [1] lists a series of task-related, hardware-related and engineering features required in tactile skin for effective utilisation, which are summarised in Fig. 1. Usually, the design of a tactile sensing skin involves some trade-offs between the different features.

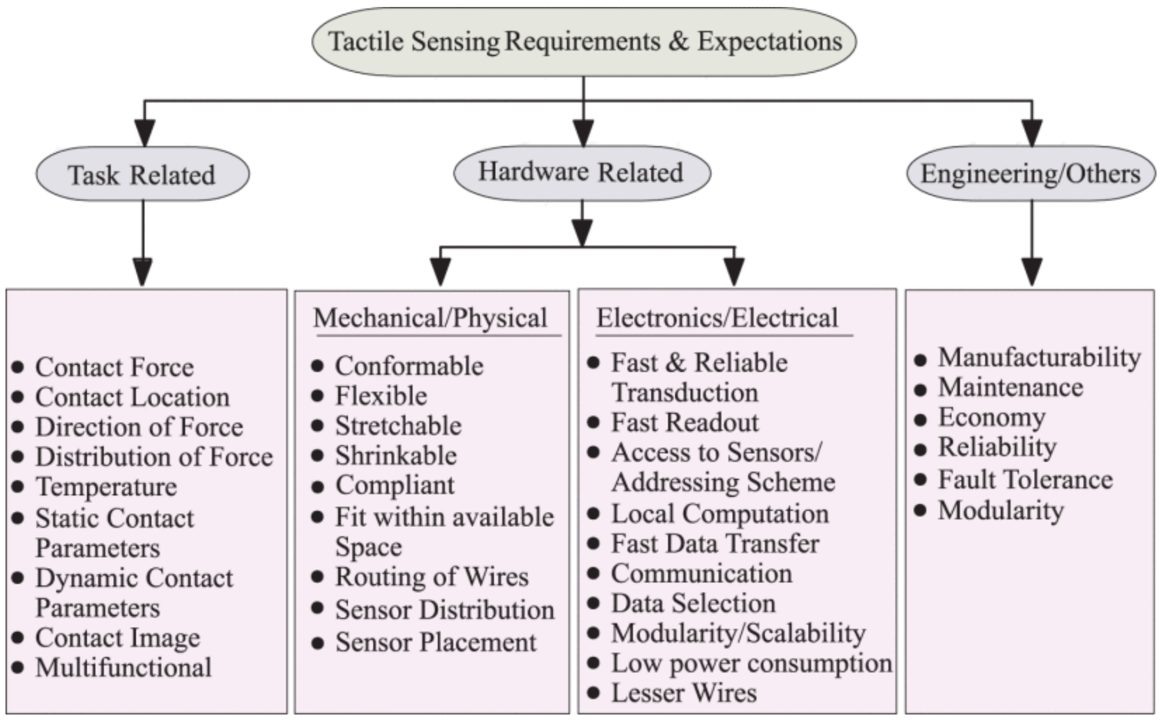


Figure 1: Tactile sensing design features: The possible features to consider for effective tactile sensor designing [1].

Tactile skin can be used in robots as a large array of sensors and other electronics embedded within flexible, stretchable and bendable substrates [1]. Like humans, robots could comprehend behaviours of real-world objects based on their weights and stiffness. Robotic tactile sensing can be task-based (action or perception) or location-based (intrinsic or extrinsic) [2]. For manipulative action tasks, tactile information can be used as a control parameter [3] - [5]. The information can include estimating contact points, measuring static forces to determine surface normal, curvature and slips [6]. There are various kinds of transduction methods used by robotic tactile sensors [2].

Piezo-resistive

Resistive sensors use piezo-resistance to determine changes in resistance with applied force at contact location and are used in anthropomorphic hands [7]. The force-sensing resistor (FSR) which is widely used in pointing and position sensing devices like joysticks, are based on piezo-resistive sensing [2]. Piezo-resistive tactile sensing is also used in microelectromechanical systems (MEMS) and silicon-based tactile sensors [8] - [9]. FSRs are of low cost, low noise, good sensitivity and simple electronics, but requiring manual assembly, stiff backing, nonlinear response and large hysteresis are drawbacks [2].

Tunnel effect

Tunnel effect sensors use quantum tunnel composites (QTC) and are also used in robot hands [10] - [11]. From an insulator, QTC can transform into a metal-like conductor when deformed by twisting, compressing or stretching [2]. A highly sensitive sensor based on electron tunnelling directly converts stress into electroluminescent light and modulates local current density, both being linearly proportional to local stress [12]. The sensor has a better spatial resolution than human fingertips with a thin film and having semiconducting nanoparticles, but charge-coupled device (CCD) camera adds to sensor size which makes it difficult for robot integration [2].

Capacitive

Capacitive sensors can be very small, which enables the arranging of dense arrays of tactile pixels or taxels [13] – [14]. The sensors on the array can be very sensitive and have a spatial resolution ten times better than humans [13]. An array of sensors can couple to object using little fibre brushes [14]. Capacitive sensors read digitised data corresponding to capacitance changes at contact points of applied force [15] and can be very sensitive, but severe hysteresis and stray capacity are drawbacks [2].

Optical

Optical sensors use light intensity changes at media of different refractive indices, to measure normal forces [16]. Three axial optical sensors can measure normal and shear forces [17]. Optical sensors are characteristically immune to electromagnetic interference, are flexible, fast and sensitive, but are quite bulky and can cause signal distortion because of light loss [2].

Ultrasonic

Ultrasonic sensors use acoustics from microphones to detect surface noise caused by motion and an array of polyvinylidene fluoride (PVDF) polymer can sense contact events from ultrasonic emissions to effectively detect slip and surface roughness during movement [18]. Ultrasonic sensors have rapid dynamic response time and good force resolution, but are difficult to miniaturise the circuits [2].

Piezoelectric

Piezoelectric sensors use materials like PVDF and can generate charge proportional to applied force. Although quartz and ceramics (PZT) have better piezoelectric properties, PVDF are preferred due to its flexibility, chemical stability and workability [19], but temperature sensitivity is a drawback [2].

Magnetic

Magnetic sensors use changes in magnetic flux density upon applied force. Hall-effect devices [20] – [21] or magneto-resistive devices [22] can measure the flux changes. High sensitivity, good dynamic range, no mechanical hysteresis, linear response and physical robustness are great advantages for magnetism-based sensors. However, applications require non-magnetic mediums [2].

Therefore, if the applications do not involve being in proximity with other magnetic objects, the ideal choice for tactile sensing are magnetic sensors due to their characteristic qualities of high sensitivity, dynamic range, no mechanical hysteresis, linear response and physical robustness. These attributes enable efficient tactile sensing applications.

B. Related Work

The very first prototypes of Hall-effect based tactile sensors were introduced in [23] [24], but were only preliminary and lacked characterisation. Using the MLX90393 triaxis® micro-power magnetometer (magnetic field sensor) integrated chip from Melexis, a soft skin sensor prototype was developed and included its characterisation [25]. Further work resulted in implementing distributed tactile sensing using this sensor chip to cover robot fingers and phalanges [26] [27]. Therefore, distributed 3-axis force sensing could represent different object shapes [27]. Earlier work was only able to measure 1-axis force [21] [28]. A three dimensional magnetic field measurable tactile Hall sensor was developed for medical diagnostics, but was not used for distributed sensing [29]. Distributed sensing is often incorporated in robots for their interaction with the external environment. A robot covered with strain gauge based tri-axial skin sensors was developed in [30], but distributed sensing may not be possible because of sensor size. Another robot, covered with soft skin sensors, was developed in [31], but a large number of sensing elements and electronics needed to be used, only to measure normal force. Capacitive soft skin sensors were used to cover another robot and used less wires by adding capacitance to digital converter, but was again only able to measure normal force [32]. Soft spherical tactile sensors were tested on robots to measure three dimensional forces in [33], but the sensor design imposed minimum size constraints. A tri-axial tactile distributed soft skin system was developed in [34]. This system, installed on a humanoid robot, could measure normal and shear forces with sub-centimetre spatial density, digital output and minimal electronics [34].

Although a lot of work has been carried out on the design and development of tactile sensors, not a lot of exploration was done on the wireless data collection and visualisation of these sensing modules. By incorporating wireless technologies, the long wires used for these systems can be reduced and data can be collected on a gateway device such as a smartphone. Nevertheless, some wireless tactile sensors were indeed developed in different settings, but proved to be preliminary and contained limitations.

Wireless tactile sensing module using stress-sensitive ceramic resonator chips implanted in tough and elastic skin was developed, where resonant frequency is read out by a ground coil with wireless coupling [35] but this type of method imposes challenges relating to distributed sensing, crosstalk, multidimensional sensitivity and practical design. Whereas, high precision tasks like medical surgery require tactile sensing feedback. For example, Peter et al. [36] developed a master-slave system to use with instrumented surgical gloves where the need for the wireless aspect was emphasised, but their neural network system was not implemented on a wireless system. Pai and Rizun [37] developed a wireless hand-held texture sensor that can measure surfaces, but could only sense single axis force and had device miniaturisation limitations. Makino et al. [38] developed a wireless piezoelectric vibration sensor which could be attached on fingernails and detect touch-based sounds, but it had an impractical design and limited applications. Wireless vibration-based tactile sensing was also demonstrated using a waist-worn belt consisting of a microcontroller [39]. Stan et al [40] developed a wireless haptic actuator system using a microcontroller to assist visually impaired people, but is complex and the applications were not extensively studied. Wireless haptic feedback gloves with vibration sensors can be used to assist the navigation of visually impaired people [41]. Tissue and organ palpation was achieved using a wireless magnetic device in [42], but the preliminary study was not explored in a realistic setting and was limited to motion in a single axis. A wireless tactile sensing data glove design was introduced in [43], but with complex and outdated electronic components. An inductive tactile sensor with deformable polymer could sense wirelessly due to magnetic coupling in [44], but could measure single axis force with a complex design. Human prostheses is another area where tactile sensing can provide its benefits and an artificial skin system with tactile sensing modules communicating over wireless sensor networks was proposed in [45], but was not implemented on real patients. A wireless disease diagnostic bio-sensor system was proposed in [46], but the design was complex and expensive. Tactile sensing is also popular in humanoid robots such as in [47], where wireless resistive-based sensing was used, but only measured single axis force. A wireless vibration-based tactile display that could operate independently was proposed in [48], but force-feedback experiments were carried out of single axis. A strain gauge wireless measurement system was developed in [49] that can diagnose structures, but the sensor is not feasible for small and soft touch applications. A wireless stylus using vibration and impact for sensing was designed in [50], but such tactile sensing has limited applications.

McDaniel et al. [39] stated that, “Our sense of touch is largely underutilized - compared to vision or hearing - as a communication channel in today's computer interfaces. This is true even when our skin, given its size, spatial acuity and temporal acuity, has shown immense potential as a communication modality.” This statement suggests that, from an evolutionary perspective, human tactile perception could be utilised more like its other senses. Keeping this in mind, tactile sensing technologies for robotics and other devices have huge potential.

The recent evolution of the Internet of Things has been fed from the exponentially progressing domains of big data, artificial intelligence, cloud computing, data analytics and miniaturised sensing technologies. Tactile Sensing is undoubtedly an increasingly useful and popular research topic due to the current advancements in electronics, robotics and sensors. Developing wireless capabilities reduce the bulky wires, to provide a detached, non-invasive and remote platform for efficient and effective sensing. So far, a miniature magnetic tactile sensor with soft skin that can measure forces in three axes and also gather data in real-time wirelessly, has not been implemented. Such a tactile sensor can be extremely useful in multitudes of applications across different industries.

II. COMPONENTS

This section describes the individual components used in the project, i.e., the sensing chip, manufacturing process of the soft shape for sensor, criteria for selection of the microcontroller, communication protocol used by sensor, how the circuit is setup, power source for microcontroller and manufacturing process of the housing shape.

A. Sensing Chip

The goal of this project was to provide a wireless tactile sensing solution that uses minimal wiring and low-cost components for the circuitry. Along with these, the size of the sensing module was also important, as smaller units can be integrated more conveniently in a wide range of applications. In the current market, Melexis manufactures a Hall-effect magnetometer sensor, MLX90393, which can measure normal and shear forces in three axes [51]. Previous works were carried out using these specific sensor chips in different arrangements and robotics applications [21], [25] – [28], [34]. These MLX90393 sensor chips are low-cost of £1.50 per chip, and small size of 3x3x1 mm dimensions. They produce 16-bit output corresponding to the magnetic flux density along the three X, Y, Z axes. If a small permanent magnet is placed on top of the sensor, the chip can measure the changes in magnetic flux of the magnet upon its movements, which can be triggered by external tactile contact.

A rigid custom Printed Circuit Board (PCB) of dimensions 9.5 x 7 x 1 mm incorporating the MLX90393 sensor, was developed at INESC MN in Portugal by Pedro Ribeiro; previous work also involved the same chip mounted on flexible PCB and integrated on a robot [52].

The rigid custom PCB is small in size and has minimal wiring requirements. It uses I^{2}C protocol to produce digital outputs and can be directly connected to a microcontroller using the four wires, VCC, GND, SDA and SCL. Fig. 2 shows the connections of the custom-made PCB with the MLX90393 chip mounted on it:

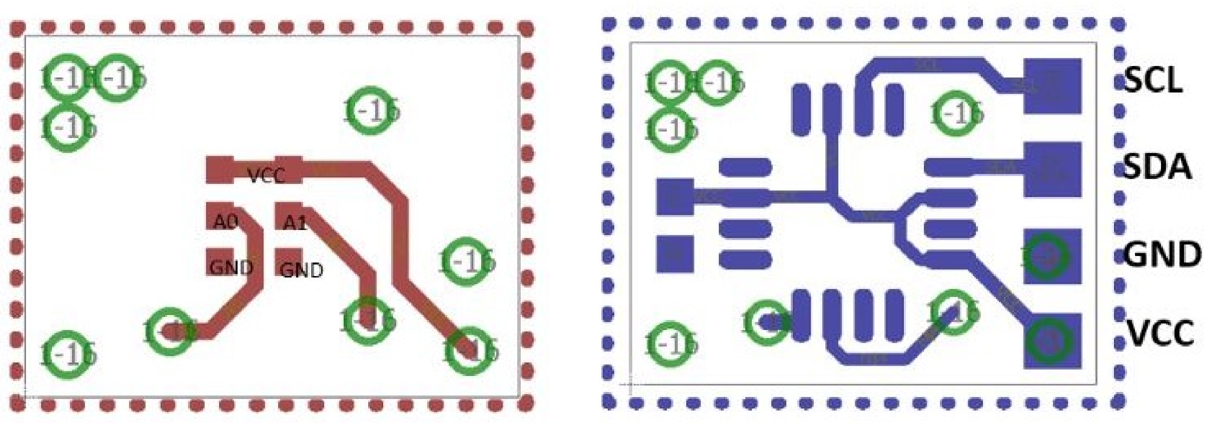


Figure 2: The PCB schematics: The connections at the front (indigo) and at the back (maroon) from the integrated sensor chip to the pads for I^{2}C wiring are shown. The pads at the back of the PCB can be used to change the I^{2}C address of the sensor by shorting connections.

B. Soft Shape Manufacturing

In order to manufacture the soft outer shape of the tactile sensing modules, 3D printing technology is used. This simplifies the process to create precise and efficient shapes. The hardware used to manufacture the shape was Form Labs Form 2 Printer and its specifications [66] are as follows:

- Technology: Stereolithography (SLA)

- Laser Spot Size: 140 microns
- Laser Power: One 250 mW laser
- Build Volume: 14.5 x 14.5 x 17.5 cm
- Layer Thickness: 25 – 300 microns
- Printer Dimensions: 34.5 x 33 x 52 cm
- Weight: 13 kg
- File Types: STL and OBJ files input, FORM file output
- Operating Temperature: Auto-heats to 35 °C

The software used to design the shape was Autodesk Fusion 360. The dimensions of the 3D shape are 14 x 11 x 6 mm. The outer rectangular area is 154 mm^2 and the inner rectangular area is 66.5 mm^2. The diameter of the magnet placeholder is 3 mm and height is 1 mm. There is an air gap of 3.5 mm to allow the deformation of the shape during the movement of the magnet. The four cylindrical holes for the wires have diameters of 1.5 mm each.

The first step to design this soft shape model on Fusion 360 is to draw the outer rectangle of dimensions 14 x 11 mm. Then, extrude the rectangle to a height of 6 mm to create a solid cuboid shape. At the bottom rectangular side of the shape, draw the inner rectangle of dimensions 9.5 x 7 mm and extrude upwards to a height of 4.5 mm, which will leave a hollow cuboid to incorporate the placement of the sensor PCB and the air gap. At the top rectangular side of the shape, draw a circle of diameter 3 mm and extrude it by a height of 1 mm to leave a space for the magnet. At the bottom right-side of the shape, draw four circles of diameters 1.5 mm each and extrude inside into the hollow chamber to provide gaps for the four wires to come out from the sensor PCB. The following four images show the top, bottom, side and isometric views of the 3D design of the soft shape (Fig. 3):

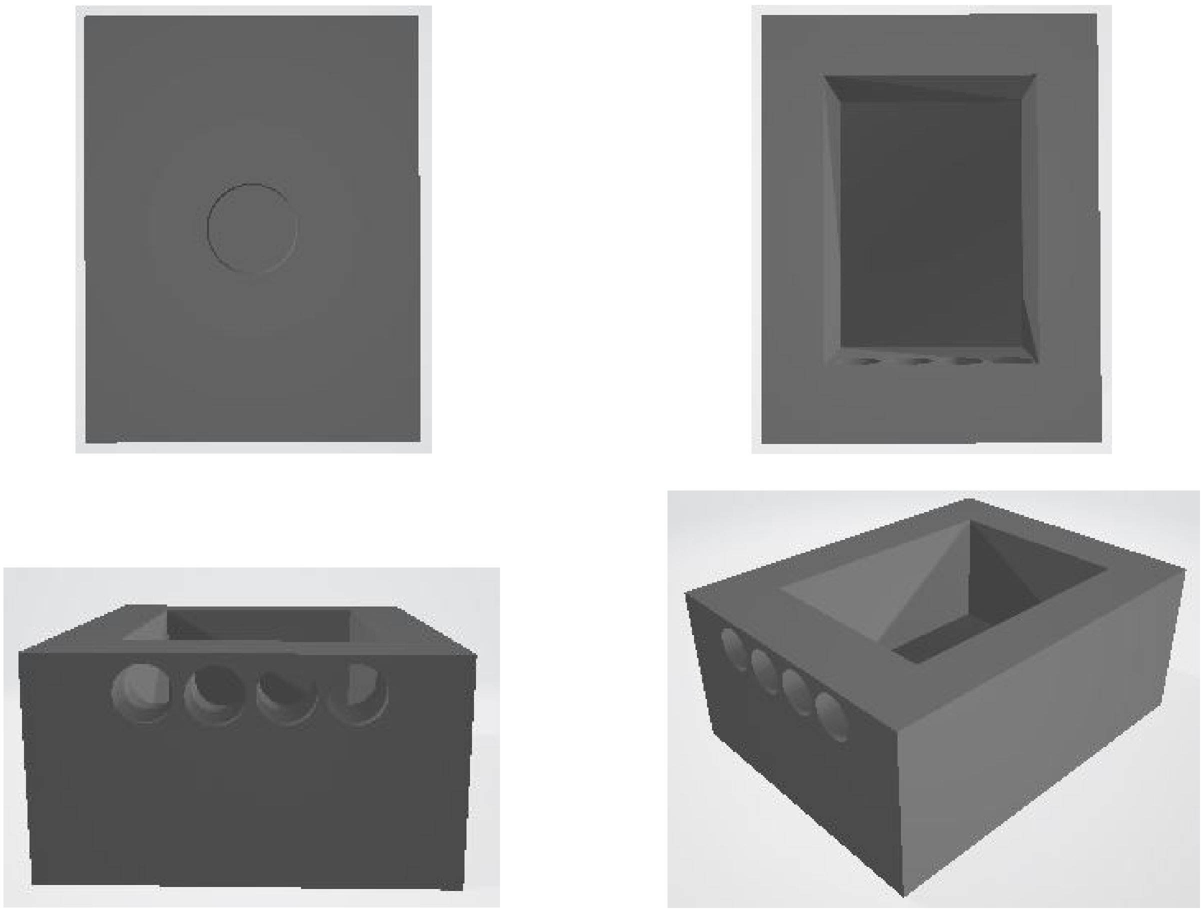


Figure 3: Soft shape model: The top view (top-left image) shows a placeholder for the magnet. The side view (bottom-left image) shows four cylindrical cut outs for the soldered wires to come out. The bottom view (top-right image) shows a placeholder for the PCB with integrated sensor chip. The isometric view (bottom-right image) shows the overall 3D structure from the bottom.

The 3D printer had a various range of material choices to select from. Based on the requirements of an ideal tactile sensing shape, the material needed to be flexible and soft. The characteristics of the materials or resins, along with images [53] of example shapes using these, are listed in Tab. 1:

Table 1: Material choices for SLA 3D printing		
Type	**Characteristics**	**Example Images**
Grey Pro Resin (for Versatile Prototyping)	This material offers high precision, high heat resistance, low creep and moderate elongation. It is great for modelling concepts and functional prototyping, especially for parts that need to be handled repeatedly. It is ideal for fit and form testing; injection moulded product prototypes; mould masters for plastics and silicones; jigs and fixtures for manufacturing [53].	
Rigid Resin (for Stiffness and Precision)	This material is reinforced with glass in order to offer very high stiffness and polished finish. It is highly resistant to deformation over time and is great for printing thin walls and features. It is ideal for turbines and fan blades; jigs, fixtures, tooling; manifolds; electrical casings and automotive housings [53].	
Durable Resin (for Low Friction and Wear)	This material offers low modulus, high elongation and high impact strength. It produces parts with a smooth, glossy finish and high resistance to deformation. It is used for applications requiring minimal friction. It is ideal for consumer packaging; bushings	

	and bearings; snap fits and flexures; living hinges [53].	
Tough Resin (for Rugged Prototyping)	This material offers balance between strength and compliance. It is ideal for sturdy prototypes; interference and press fits; assemblies [53].	
Flexible Resin (for Hard Flexible Parts)	This material is suitable for rigid flexible parts with matte-black soft-touch finish. With 80A Shore durometer, it is close to rubbers used to producing shoe soles or tire treads. It is ideal for cushioning and damping; wearables and consumer goods prototyping; handles, grips and over moulds [53].	
High Temp Resin (for High Thermal Stability)	This material offers a heat deflection temperature of 238 °C @ 0.45 MPa. It is ideal for hot air, gas and fluid flow; heat resistant mounts, housings and fixtures; moulds and inserts [53].	
Elastic Resin (for Soft Flexible Parts)	This material is suitable for prototyping parts normally produced with silicone. It is used for applications that requires bending, stretching, compressing, and holding up to repeat cycles without tearing. It is the softest among the choice of resins with 50A Shore	

	durometer, along with high elongation and energy return. It is ideal for wearables and consumer goods prototyping; medical models and devices; compliant features for robotics; special effects props and models [53].	

Choosing the Elastic Resin seemed ideal due to its high elongation and soft silicon-like properties. Fig. 4 shows a comparison of elongation against ultimate tensile strength for the various resin choices:

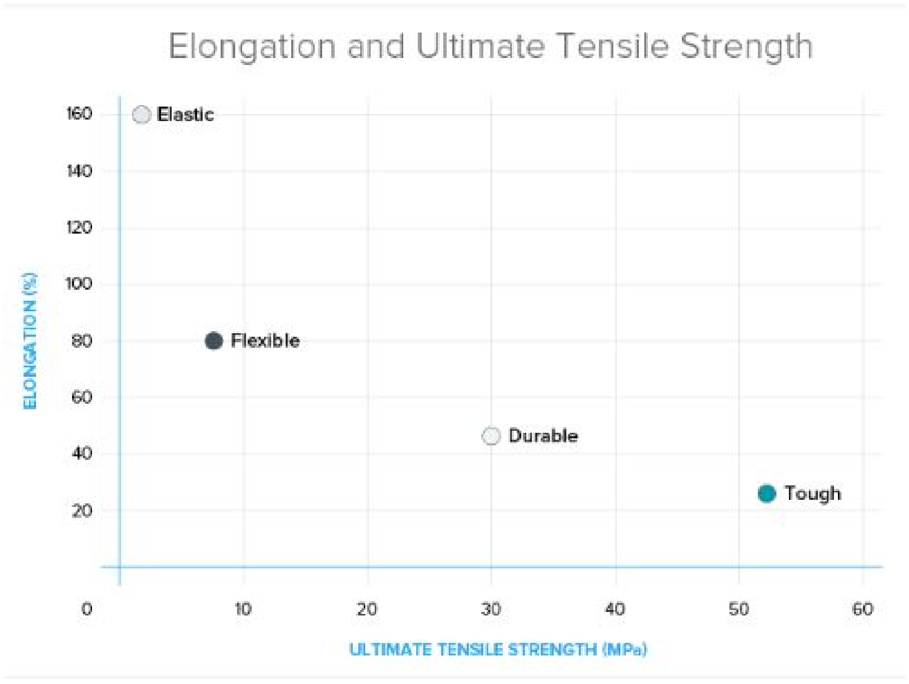


Figure 4: Resin comparison: Elongation (%) versus Ultimate Tensile Strength (MPa) of Elastic, Flexible, Durable and Tough resins for SLA 3D prints [53].

C. Microcontroller

In the ever-changing world of technology, the exponentially accelerating progress in the development of electronics has led to the invention of powerful computers as small as the palm of our hands. A microcontroller is essentially an integrated circuit that can singularly act as a tiny computer or controller. Therefore, a microcontroller contains processing units (CPUs), memory (ROM/RAM) and input/output peripherals, just like a regular computer. Along with a small size, microcontrollers are usually also inexpensive. As a result, it is a very economical choice in the field of the internet of things, and can enable the development of embedded systems that can interface with the external world through sensors and actuators.

Modern day microcontrollers are being developed by many different manufacturers all over the world. While each are unique for their own characteristics, the choice completely depends on the application. From the recent market, several different microcontrollers are listed in Tab. 2, along with their prices and some of their basic features [54]:

Table 2: Choices of microcontrollers

Microcontroller	**Price (£)**	**Features**
CHIP	7	1GHz processor, 8 GPIO pins, Bluetooth 4.0 and Wi-Fi, 4GB storage
Mediatek LinkIt One	46	260MHz speed, GPS, GSM, GPRS, Wi-Fi, and Bluetooth

Particle Photon	15	120Mhz, five analogue pins and eight digital pins, Wi-Fi, 1MB flash and 128KB RAM
Tessel	35	580MHz, Wi-Fi and Ethernet ports, 64 MB DDR2 RAM & 32 MB Flash
Udoo Neo	51	1GHz processor, Bluetooth 4.0 and Wi-Fi, 1GB RAM
LightBlue Bean	23	BLE, 8 GPIO pins, 32KB Flash memory and 2KB SRAM
Intel Edison	55	Wi-Fi, Bluetooth 4.0, 1GB DDR and 4GB flash memory, 20 digital input/output pins
Raspberry Pi	27	1.2 GHz processor, Wi-Fi and Bluetooth 4.1, 40 GPIO pins, 1GB RAM
Arduino Uno	18	14 digital input/output pins and 6 analogue inputs, 32KB flash memory

The aforementioned microcontroller may be decent choices for certain applications and projects. However, when choosing the ideal candidate, other factors come into consideration such as availability, cost, resources, efficiency, size, compatibility and so on. Therefore, the choice of microcontroller for this project is the low-cost and highly popular ESP32 with integrated Wi-Fi and Bluetooth technologies.

The predecessor of ESP32, which is called ESP8266, is still among the most popular microcontroller choices. In summary, both the chips have a 32-bit processor and Wi-Fi, but the successor ESP32 furthermore contains Bluetooth, SRAM, Flash, additional GPIO (general purpose input/output) pins, Ethernet MAC interface, CAN, sensors (touch, temperature and hall effect) [55]. Hence, the newer ESP32 is the most ideal choice between all the listed microcontrollers.

ESP32 is a single 2.4 GHz Wi-Fi and Bluetooth combination chip designed with the TSMC (manufacturer) ultra-low-power 40 nm technology and is developed by Espressif Systems, a Shanghai-based Chinese company [56]. ESP32 has compatibility to be programmed on several platforms, including the most popular Arduino IDE. It is a system-on-a-chip microcontroller that is both low-cost and low-power. It can perform as a slave device to another host MCU and also act as a complete standalone system. ESP32 can interface with other systems through its SPI or I^{2}C interfaces. It is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C. ESP32 is highly-integrated with in-built antenna switches, power amplifier, low-noise receive amplifier, RF balun, filters, and power management modules [57]. It can be used with minimal Printed Circuit Board (PCB) requirements. ESP32 is engineered for ultra-low power consumption and is ideal for the Internet of Things applications with mobile devices and wearable electronics. It has state-of-the-art features like fine-grained clock grating, different power modes and dynamic power scaling. To save power, ESP32 is woken up periodically and only when a specified condition is detected. Low-duty cycle enables minimising the amount of energy that the chip uses up. The output of the power amplifier is adjustable, thus contributing to an optimal trade-off between communication range, data rate and power consumption [56].

ESP32 can switch between various power modes using advanced power management technologies. The various power modes are as follows [56]:

- ✓ Active mode: The chip radio is powered on. The chip can receive, transmit, or listen.
- ✓ Modem-sleep mode: The CPU is operational and the clock is configurable. The Wi-Fi/Bluetooth baseband and radio are disabled.
- ✓ Light-sleep mode: The CPU is paused. The RTC memory and RTC peripherals, as well as the ULP co-processor are running. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
- ✓ Deep-sleep mode: Only the RTC memory and RTC peripherals are powered on. Wi-Fi and Bluetooth connection data are stored in the RTC memory. The ULP co-processor is functional.
- ✓ Hibernation mode: The internal 8-MHz oscillator and ULP co-processor are disabled. The RTC recovery memory is powered down. Only one RTC timer on the slow clock and certain RTC GPIOs are active. The RTC timer or the RTC GPIOs can wake up the chip from the Hibernation mode.

When Wi-Fi is enabled, the chip switches between Active and Modern-sleep modes. During Modern-sleep mode, the CPU frequency changes automatically and depends on CPU load and used peripherals. In Deep-sleep mode, when ULP co-processor is powered on, peripherals like GPIO and I^2C can operate [56].

ESP32 implements a TCP/IP and full 802.11 b/g/n Wi-Fi MAC protocol. It supports the Basic Service Set (BSS). STA and SoftAP operations under the Distributed Control Function (DCF). Power management is handled with minimal host interaction to minimize the active-duty period [56].

The ESP32 Wi-Fi Radio and Baseband support the following features [56]:

- ✓ 802.11 b/g/n
- ✓ 802.11n MCS0-7 in both 20 MHz and 40 MHz bandwidth
- ✓ 802.11n MCS32 (RX)

- ✓ 802.11n 0.4 µs guard-interval
- ✓ 802.11 n (2.4 GHz), up to 150 Mbps of data rate
- ✓ Receiving STBC 2×1
- ✓ Up to 20.5 dBm of transmitting power
- ✓ Adjustable transmitting power
- ✓ Antenna diversity

ESP32 supports antenna diversity with an external RF switch. One or more GPIOs control the RF switch and selects the best antenna to minimize the effects of channel fading [56].

The ESP32 Wi-Fi MAC applies low-level protocol functions automatically which are as follows [56]:

- ✓ WMM
- ✓ TXOP
- ✓ CCMP (CBC-MAC, counter mode), TKIP (MIC, RC4), WAPI (SMS4), WEP (RC4) and CRC
- ✓ TX/RX A-MPDU, RX A-MSDU
- ✓ RTS protection, CTS protection, Immediate Block ACK
- ✓ Defragmentation
- ✓ Automatic Beacon monitoring (hardware TSF)
- ✓ 4 × virtual Wi-Fi interfaces
- ✓ Simultaneous support for Infrastructure Station, SoftAP, and Promiscuous modes. When ESP32 is in Station mode, performing a scan, the SoftAP channel will be changed.

ESP32 has integrated Bluetooth link controller and baseband that carries out baseband protocols and other low-level link routines like modulation/demodulation, packet processing, bit stream processing and frequency hopping [56].

The ESP32 Bluetooth Radio and Baseband support the following features [56]:

- ✓ Class-1, class-2 and class-3 transmit output powers, and a dynamic control range of up to 24 dB
- ✓ π/4 DQPSK and 8 DPSK modulation
- ✓ High performance in NZIF receiver sensitivity with over 97 dB of dynamic range
- ✓ Class-1 operation without external PA
- ✓ Internal SRAM allows full-speed data-transfer, mixed voice and data, and full piconet operation
- ✓ Logic for forward error correction, header error control, access code correlation, CRC, demodulation, encryption bit stream generation, whitening and transmit pulse shaping
- ✓ ACL, SCO, eSCO and AFH
- ✓ A-law, µ-law and CVSD digital audio CODEC in PCM interface
- ✓ SBC audio CODEC
- ✓ Power management for low-power applications
- ✓ SMP with 128-bit AES

The Bluetooth interface provides UART HCI interface up to 4 Mbps, PCM/I^{2}C audio interface, and SDIO/SPI HCI interface. The Bluetooth stack is compliant with Bluetooth v4.2 BR/EDR and BLE specifications [56].

The link controller operates in three major modes: standby, connection and sniff. The technical features [56] of Classic Bluetooth are:

- ✓ Device Discovery (inquiry, and inquiry scan)
- ✓ Connection establishment (page, and page scan)
- ✓ Multi-connections
- ✓ Asynchronous data reception and transmission
- ✓ Synchronous links (SCO/eSCO)
- ✓ Master/Slave Switch

- ✓ Adaptive Frequency Hopping and Channel assessment
- ✓ Broadcast encryption
- ✓ Authentication and encryption
- ✓ Secure Simple-Pairing
- ✓ Multi-point and scatternet management
- ✓ Sniff mode
- ✓ Connectionless Slave Broadcast (transmitter and receiver)
- ✓ Enhanced power control
- ✓ Ping

The technical features [56] of Bluetooth Low Energy are:

- ✓ Advertising
- ✓ Scanning
- ✓ Simultaneous advertising and scanning
- ✓ Multiple connections
- ✓ Asynchronous data reception and transmission
- ✓ Adaptive Frequency Hopping and Channel assessment
- ✓ Connection parameter update
- ✓ Data Length Extension
- ✓ Link Layer Encryption
- ✓ LE Ping

Therefore, ESP32 is an ideal choice for this project or any other Internet of Things applications. In this project, the integrated Wi-Fi technology of ESP32 is utilised, but in future work the Bluetooth functionality could also be useful.

D. I^{2}C Protocol

I^{2}C or IIC (Inter-Integrated Circuit) is a synchronous, serial protocol invented by Philips Semiconductor in 1982. Most current manufacturers use I^{2}C in their developments of integrated chips. I^{2}C can be used to connect low-speed devices such as microcontrollers and other input/output peripherals in embedded systems.

I^{2}C has a very simple usage design with only two wires for bus connectivity, serial clock (SCL) and serial data (SDA). Multiple I^{2}C buses of varying voltages can be connected using I^{2}C level shifters, as well. I^{2}C protocol can have multiple master devices and multiple slave devices. Each slave device needs to have a unique 7-bit address on the bus. Some devices can also have 10-bit addresses, depending on their specifications. The master device does not need an address as it triggers the clock through SCL to identify the I^{2}C slave devices. I^{2}C communication takes place serially on the bus using packets of 8 bits or bytes [58].

I^{2}C protocol is a very popular choice for digital electronics because an I^{2}C bus can communicate with several devices. ESP32 has I^{2}C communication support and in fact, contains dedicated hardware for I^{2}C bus. In theory, only two free pins are required on the microcontroller to connect an infinite number of I^{2}C slave devices, as long as they each have a unique address on the connected bus. The default I^{2}C pins on an ESP32 board is GPIO21 and GPIO22 for SDA and SCL connections respectively. The I^{2}C controller on ESP32 supports both standard mode of 100 kbps, and fast mode of 400 kbps. ESP32 also supports both 7-bit and 10-bit addressing [59].

Being the master device, when the ESP32 sends out a start condition, SDA line will be pulled low and SCL line will be pulled high. 9-bit clock pulses will be sent over the SCL line to the slave device(s). The first seven pulses will correspond to the 7-bit address and eighth pulse is a read/write bit. If there is an active slave device on the bus matching this address, then the slave device will acknowledge by pulling SDA low on ninth pulse. SDA line will only change when the SCL line is low, during data transfer. The master device will send out a stop condition by pulling up SDA while SCL is high, to end the communication [59].

Fig. 5 shows the output pins layout of the specific ESP32 development board used in this project, among which only four pins (GPIO21, GPIO22, VCC and GND) were used to connect the I^{2}C sensor successfully:

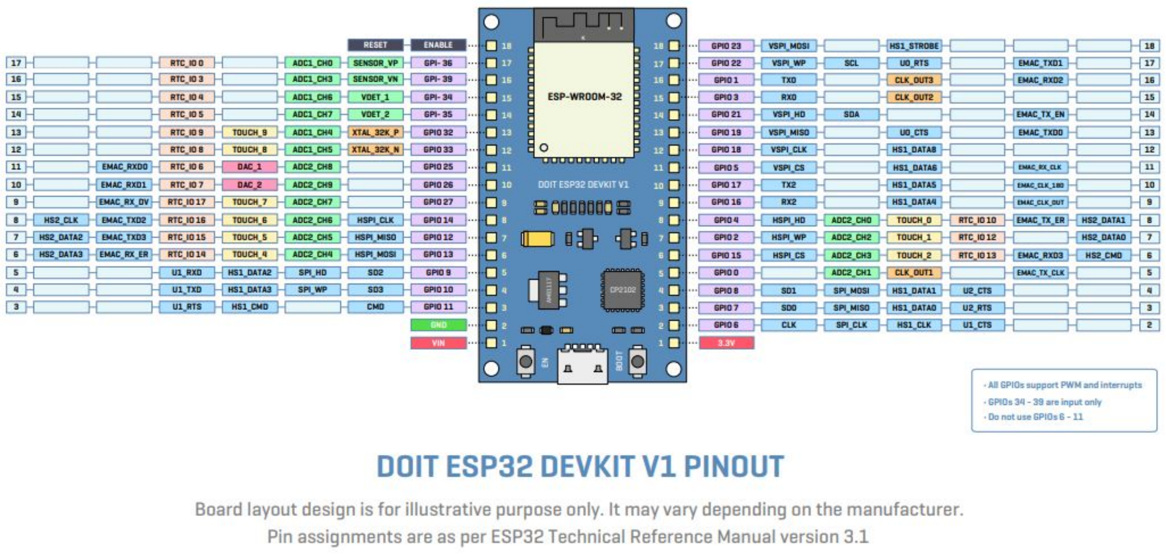
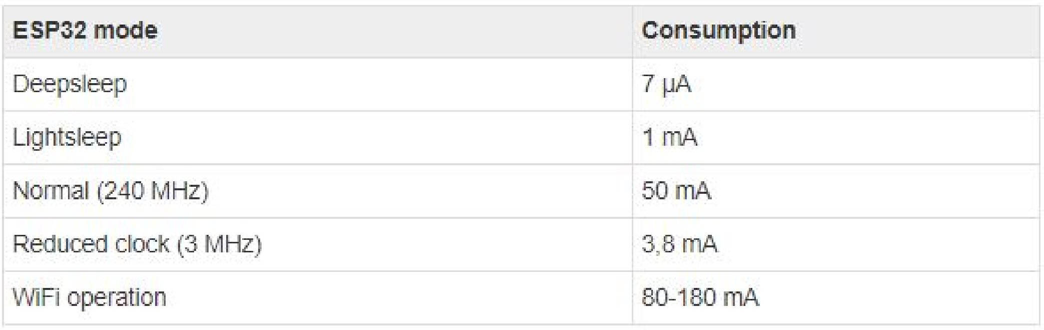

ESP-WROOM-32
DOIT ESP32 DEVKIT V1
EN
BOOT
CP2102
RESET
ENABLE
GND
VIN
3.3V
GPIO 23
GPIO 22
GPIO 21
SDA
SCL
- All GPIOs support PWM and interrupts
- GPIOs 34 - 39 are input only
- Do not use GPIOs 6 - 11
DOIT ESP32 DEVKIT V1 PINOUT
Board layout design is for illustrative purpose only. It may vary depending on the manufacturer.
Pin assignments are as per ESP32 Technical Reference Manual version 3.1

Figure 5: ESP32 pinout: The pinout layout and functionalities of the DOIT ESP32 DEVKIT V1 development board [60].

E. Power Source

ESP32 is a very energy-efficient microcontroller compared to the majority of microcontrollers in the market. It has different operation modes to utilise energy efficiency. Fig. 6 shows the recorded power consumption of the different modes of ESP32-WROOM module:



ESP32 mode	Consumption
Deepsleep	7 µA
Lightsleep	1 mA
Normal (240 MHz)	50 mA
Reduced clock (3 MHz)	3,8 mA
WiFi operation	80-180 mA

Figure 6: ESP32 modes: The power consumption values at different operating modes of ESP32 [61].

ESP32 operates at 3.3V voltage via a micro USB cable. In order to externally power the ESP32, different battery options can be considered. A comparison study identifies the different options as the power source for the ESP32 [61]. If a power bank, which uses an internal 3.7 lithium battery that converts to 5V, is connected to an ESP32, then the voltage again has to be reduced to 3.3V and is an inefficient method resulting in energy losses. Furthermore, some power banks may automatically switch off, detecting no load when ESP32 is operating in low power modes. Standard NiMH batteries have voltages of 1.2V, hence two of these fall short of the minimum required voltage of 2.55V to power ESP32, and three of these exceed the power requirement of 3.3V. One 3V or two 1.5V lithium batteries can power an ESP32 successfully. Although they provide 70% less energy compared to a lithium battery, $LiFePO_4$ batteries can also work well and also have rechargeable capability. Another option is a lithium polymer battery which provide enough voltage to power ESP32, but require voltage reduction from 4.2V to 3.3V and is not an efficient method for longevity. To summarise, lithium battery is the ideal choice to power ESP32 from weeks to years [61].

Lithium batteries have many variations and Samsung manufactures lithium-ion batteries called 25R of 18650 size. The 3.6V Samsung 25R batteries are rechargeable with a 2500mAh capacity and weight of less than 45g, and the dimensions of the battery are 18x64 mm [62].

There are specific 18650 battery holder boards to power 3.3V devices such as ESP32 in the market and the DIY More ESP32 18650 Battery Shield V3 is selected for this project. Its features include battery protection of over-charging or over-discharging, a micro USB input port to recharge battery, a switch to control USB output and a 3V USB output port. The battery shield has dimensions of 9.8x2.9 cm. LEDs indicate red when charging and green when battery is fully charged [63].

F. Stripboard Circuit

The ESP32 is the only major electronic component to be used in the circuit. Additionally, two pull-up resistors of 4.3kΩ resistance are used. One is connected between VCC and SDA, and another between VCC and SCL. There is very minimal wiring used, with only four I^2C connections (VCC, GND, SDA, SCL) soldered into the circuit. The serial data (SDA) and serial clock (SCL) wires from the sensor are connected to the default digital I^2C pins of the ESP32, which are D21 and D22. The VCC and GND wires from the sensor are connected to the VCC and GND pins of the ESP32 respectively. Finally, all the components are soldered into a thin stripboard to minimise space on the device and avoiding a bulkier breadboard. Fig. 7 shows the layout of all the electronics in the circuitry:

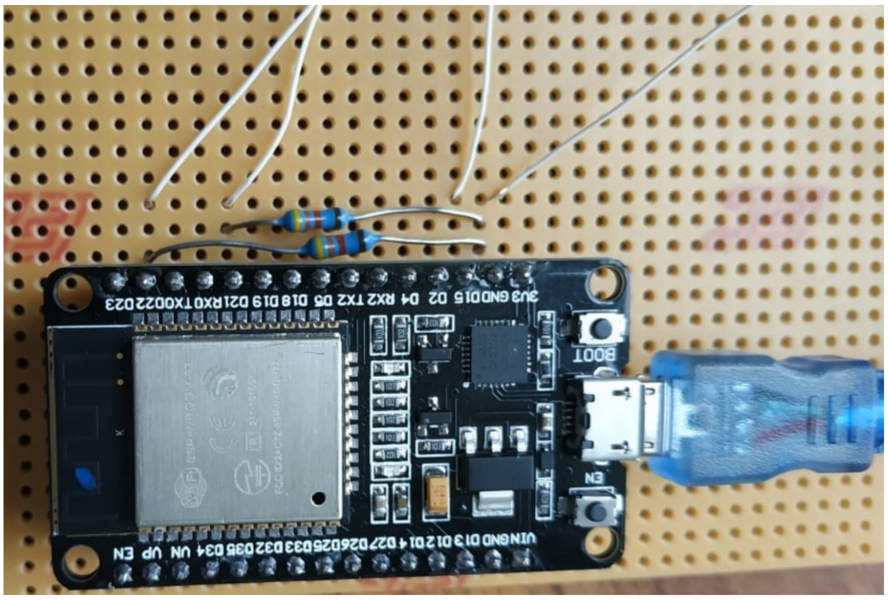


Figure 7: Stripboard circuit: The overall circuit for this project, featuring two 4.3kΩ resistors and an ESP32 connected directly to the four I^2C wires of the sensor.

G. Housing Manufacturing

To house all the components together, including the electronics, power source and sensing unit, a housing platform is designed and 3D printed. The model is designed using Autodesk Fusion 360 software and 3D printed using Ultimaker 3 hardware. The technical specifications [64] of the hardware are as follows:

- Dual extrusion build volume: 197 x 215 x 200 mm
- Assembled dimension: 342 x 505 x 688 mm
- Print technology: Fused filament fabrication (FFF)
- Weight: 10.6kg

- Maximum power output: 221W
- Nozzle diameters: 0.25 x 0.4 x 0.8 mm
- XYZ resolution: 12.5, 12.5, 2,5 micron
- Build speed: < 24 mm^3/s
- Nozzle temperature: 180 – 280 °C
- Operating sound: 50dBA
- Build plate levelling: Active levelling
- Build plate: 20 – 100 °C heated glass build plate
- Supported file types: STL, OBJ, X3D, 3MF, BMP, GIF, JPG, PNG

The material used is Ultimaker PLA (polylactic acid), which is highly versatile and comes in 11 different colours. It can print with high dimensional accuracy and quality surface finish. It can be used for a wide range of 3D printing. The material can achieve great tensile stiffness, dual extrusion and can print at high speeds. The material has characteristic properties of excellent flexural strength (103 MPa), impact strength (Izod tested to 5.1 kJ/m^2), low melting temperature (145 °C) and high hardness (83 Shore D) [65].

The design of the model requires precision of measurements in order for all the individual components to fit in their places, and to ensure that the design is as compact as possible. Since the Ultimaker 3D printer can only efficiently print open shapes, as opposed to objects with hollow chambers, the housing model had to be broken down into three parts and individually designed on Autodesk Fusion 360.

In order to design the upper body piece as shown in Fig. 8, the first step is to draw a rectangle of 180 x 100 mm and then extrude it by a height of 40 mm. Then, from the cuboid block, choose the long side and draw a rectangle of dimensions 170 x 30 mm and extrude it inwards by a height of 20 mm to create a hollow chamber for the battery shield. At the top side of the upper body piece where the sensor unit will be placed, draw four circles of diameters 3 mm each and extrude all the way to the bottom end, for the wires to go through into the soldered circuit. Also at the top side, draw and extrude two small rectangular slits of dimensions 15 x 10 x 10 mm and 10 x 10 x 10 mm into the hollow chamber for the battery shield to reserve some gaps in order to access the micro USB charging port and switch of the battery shield respectively. At the bottom side of the upper body piece, draw a rectangle of dimensions 100 x 70 mm and extrude upwards by a height of 25 mm to create a hollow chamber for the circuit placement. At the right-side of the upper body piece, draw two circles of diameters 15 mm each and extrude inwards into the hollow chambers for the battery shield and circuit respectively. These two holes enable the USB cable connection from the battery shield into the microcontroller to power it.

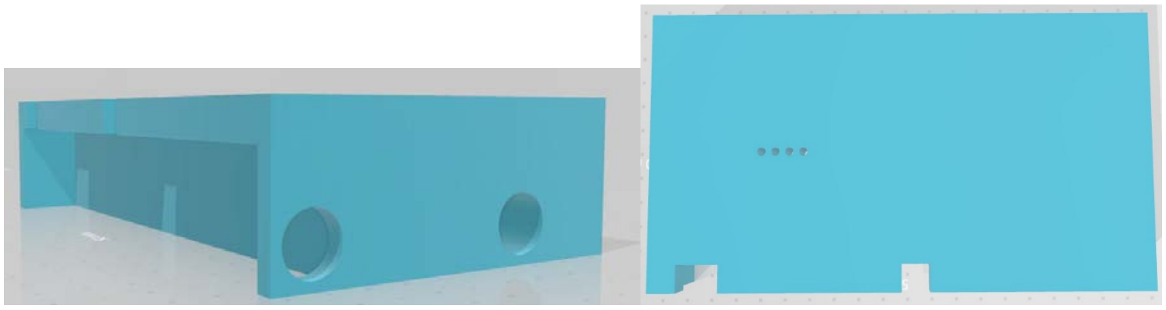


Figure 8: Upper housing piece: The left image shows the angular side view and the right image shows the top view.

In order to design the middle body piece as shown in Fig. 9, the first step is to draw a rectangle using the same dimensions of 180 x 100 mm to align with the upper body piece as these parts would be stuck together. Similarly, extrude the inner hollow chamber of dimensions 100 x 70 x 10 mm to incorporate space for the circuit placement. At the side where the hollow chamber for the battery shield is placed, draw a rectangle of dimensions 100 x 3 mm and extrude downwards by a height of 4 mm to create a slit that would keep the battery sheild standing in its place. Near the vertices, draw four circles of diameters 2.89 mm each extrude upwards by a height of 12 mm into the upper body piece, to create hollow chambers for four screws.

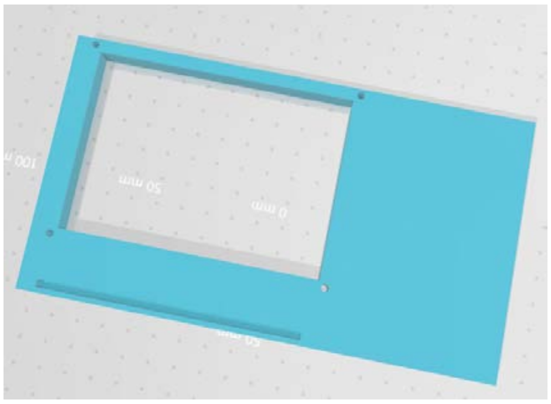


Figure 9: Middle housing piece: The top view is shown.

In order to design the bottom piece as shown in Fig. 10, draw a rectangle of dimensions 180 x 100 mm and extrude by a height of 5 mm, which will align in with the rest of the outer housing body. In the exact position where the hollow chamber for the circuit exists in the upper and middle body pieces, draw a rectangle of dimensions 100 x 70 mm and extrude upwards by a height of 2 mm, to create a lifted stage that keeps the circuit in place. This bottom piece requires to be screwed into the upper body piece so the four screw holes are in the same exact positions aligning with the upper body piece.

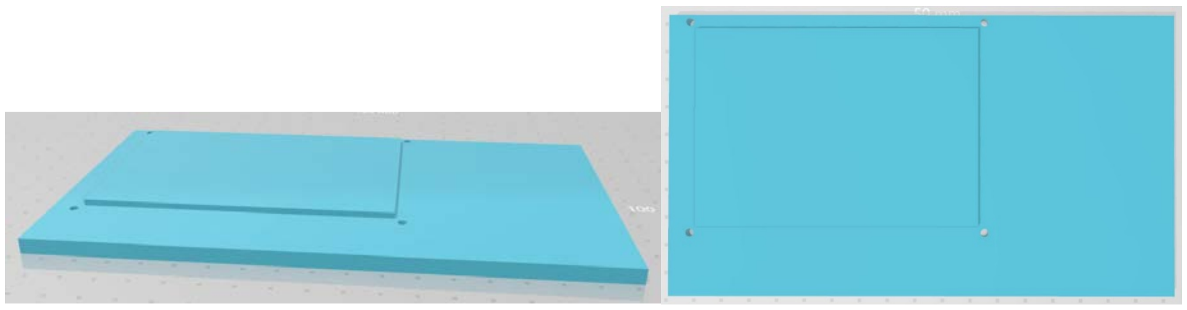


Figure 10: Lower housing piece: The left image shows the angular side view and the right image shows the top view.

Finally, the overall housing model is shown in Fig. 11, as listed below:

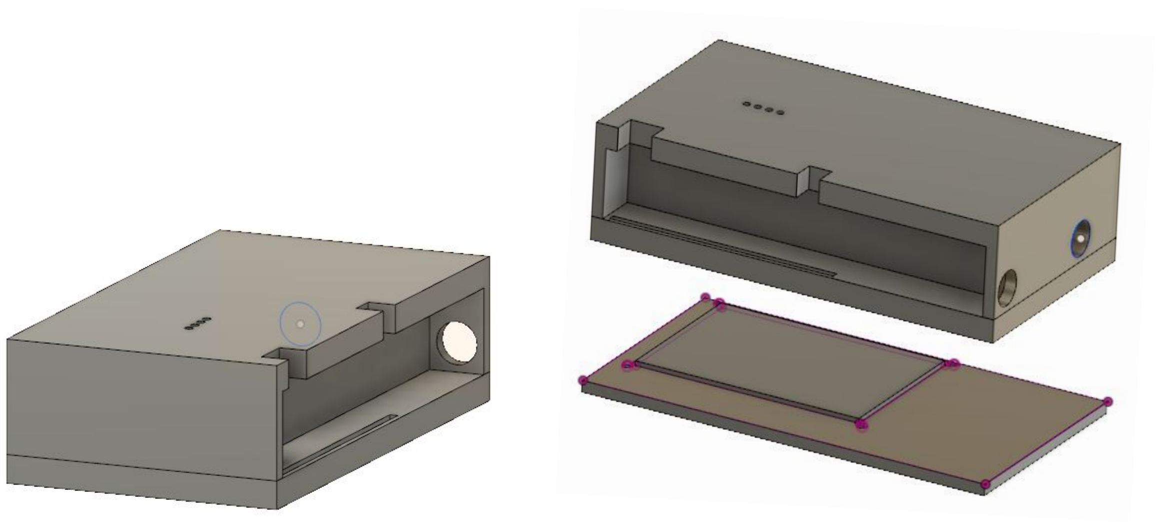


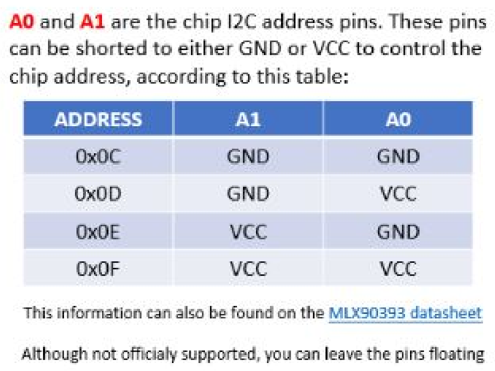
Figure 11: Overall housing model: The left image shows the overall isometric view of the housing and the right image shows the two separate pieces that are screwed together.

III. TESTING

The wireless data collection and visualisation is tested in this section. Along with this, the possibility to control multiple similar sensors using the same microcontroller is also discussed.

A. Addressing Sensors

Every slave device needs to be identified by the master device using its own unique address. In the case of this project, the microcontroller being the master device requests for the hexadecimal I^2C address of the MLX90393 tactile sensor. The custom PCBs designed for these tactile sensors have a default address of 0x0F in hexadecimal, which is 15 in decimal. In many applications, it may be useful to have multiple tactile sensors working together using the same microcontroller. Although this is very much possible since I^2C communication takes place using a common serial bus, which can connect to multiple sensors and each sensor connected to the shared I^2C bus requires its own unique address. The default address of these tactile sensors can be changed by simply shorting the pins at the back of the PCB, as suggested in Fig. 12.



A0 and A1 are the chip I2C address pins. These pins can be shorted to either GND or VCC to control the chip address, according to this table:

ADDRESS	A1	A0
0x0C	GND	GND
0x0D	GND	VCC
0x0E	VCC	GND
0x0F	VCC	VCC

This information can also be found on the MLX90393 datasheet

Although not officialy supported, you can leave the pins floating

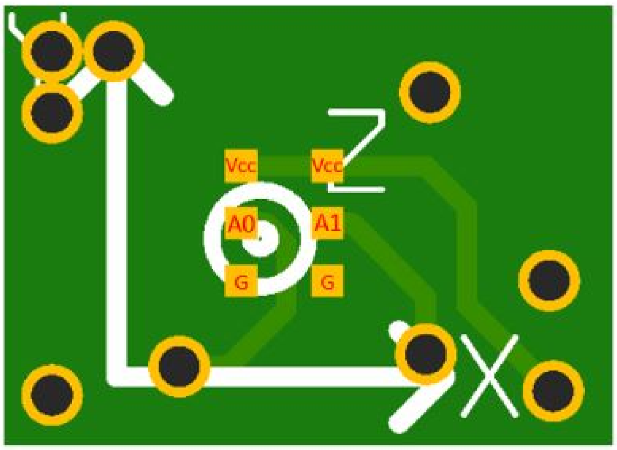
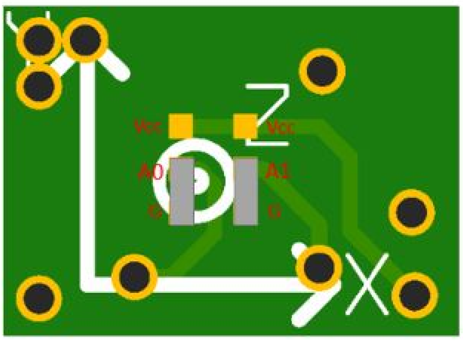


Figure 12: Addressing sensor: The back of the PCB showing the six pads which can be shorted to obtain different addresses from the default one.

For this specific custom-made PCB, only four unique addresses are possible and hence, a maximum of four sensors can be connected together within the same I^{2}C bus. If more sensors are required to be connected, then a secondary master device or microcontroller requires to be used. By default, the A0 and A1 pins are connected to VCC, which gives their default address of 0x0F in hexadecimal. However, depending on how the A0 and A1 pins are shorted, three other hexadecimal addresses of 0x0C, 0x0D and 0x0E are also possible to be assigned. To clarify things, Fig. 13 shows an example of the default I^{2}C address of the sensor being changed into a different address.

Example

If you want to assign address **0x0C** to a chip, you must short the pins (e.g. by soldering) in the board backside as shown in the image



A jumper, a soldered wire or a solder bridge can be used, as long a connection can be made between the two pins

Figure 13: Addressing example: The back of the PCB showing an example of A0 and A1 shorted to GND by soldering, which enables the chip to assign the address of 0x0C.

If the address of the sensor is unknown or once the new address of the sensor is assigned, then I^{2}C scanner program can scan the address of the connected sensor. An open-source example of an I^{2}C scanner program can be found in Appendix 1. A working program that successfully sequentially reads from two different MLX90393 sensors with different addresses on the same I^{2}C bus can be found in Appendix 2.

B. Running Program

In this project, the particular MLX90393 tactile sensor used has the default address of 0x0F in hexadecimal, as shown in the following code snippet:

```
// MLX90393 I2C Address is 0x0F(15)
#define Addr 0x0F
```

The full code can be found in Appendices 3 and 4. The main code file is named 'ESP_Web_Graphs.ino' and can be found in Appendix 3. The code in Appendix 4 is a function that is called from the loop part of the main code. The original code files can be found in supporting materials along with a README file, which contains the instructions for compiling and using it.

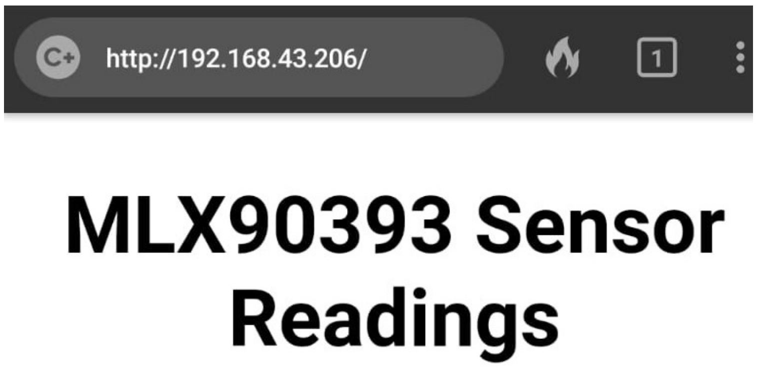
C. Web Server

The wirelessly collected data is visualised in a web server using the ESPAsyncWebServer library [67]. The web server is asynchronous and displays three graphs to plot the readings of forces in X, Y and Z axes in real-time. The graphs plot a maximum of 40 points before newer data points arrive. There is a new data point reading every 0.5 seconds, which is almost real-time.

The graphs are built using the Highcharts library [68]. A HTML file is created for the webpage displaying the three graphs. This HTML file can be uploaded into the ESP32 using the Arduino IDE File system uploader (for more information, refer to the README file in supporting materials). Two libraries need to be installed on Arduino IDE before the asynchronous server is built, ESPAsyncWebServer [67] and AsyncTCP [69] (for more information, refer to the README file in supporting materials).

The HTML file is uploaded into the ESP32 first, followed by the main Arduino sketch file. The specific ESP32 development board used in this project requires pressing down on the

Boot button while the files are being uploaded, hence when the Arduino IDE completes the compilation and shows the ‘Connecting…’ message, the Boot button of ESP32 is pressed down to complete the uploading of the files. Once all the files are uploaded, a browser is opened on the Wi-Fi display device and directed to the local IP address as shown in Fig. 14a. If the ESP32 is connected successfully with the Wi-Fi device, in this case a smartphone, then it will display the three graphs along with new data points every half a second. Arduino IDE serial monitor can show if the Wi-Fi device is connected to the ESP32, if needed. Once connected, the serial monitor will provide the local IP address and then start displaying the sensor readings.


http://192.168.43.206/

MLX90393 Sensor Readings

Figure 14a: Local IP: The Wi-Fi device displaying the local IP address.

D. Data Visualisation

The wireless tactile sensing is achieved using the ESP32 and displaying the measured sensor readings on graphs. The data points in the graphs are plotted in real-time with a small delay of half a second. There is a maximum of 40 points possible to be listed in each graph at a time. Fig. 14b shows the three graphs which display the magnetic flux changes in X, Y and Z axes, and the differences in the data points exist because the sensor was touched during those instances. Appendix 5 shows the HTML code which displays the three graphs as a webpage.

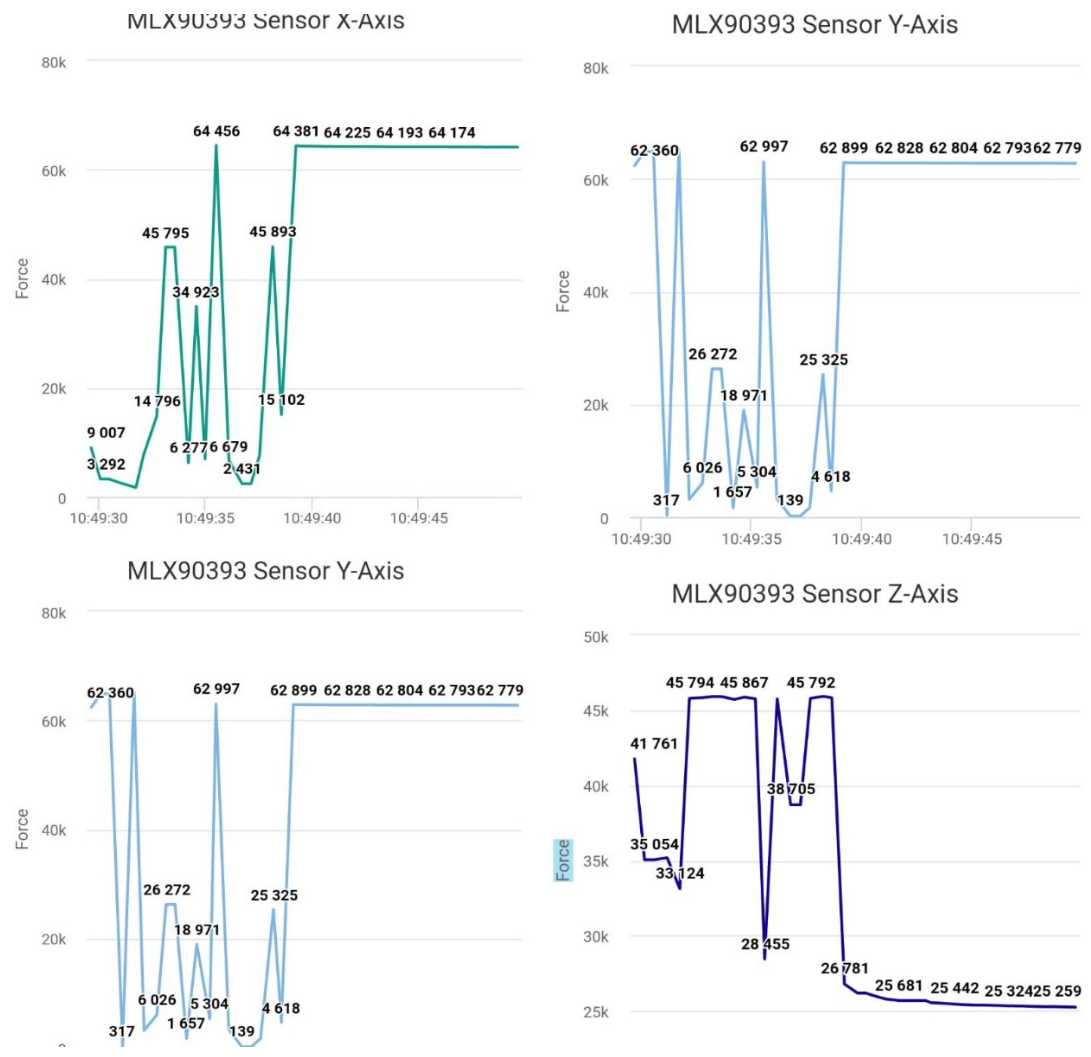


Figure 14b: Result graphs: The left screenshot shows X and Y graphs, and right screenshot shows Y and Z graphs, which show the magnetic flux changes of the tactile sensor.

The graphs can also show the exact time when each data point was captured, by selecting an individual data point as shown in Fig. 14c.

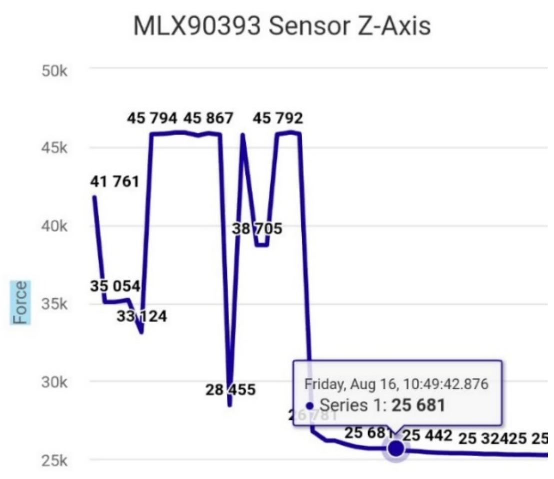


Figure 14c: Timestamp: Exact day, date and time of selected data point.

The final prototype with all its individual components, is presented in a physical housing design and is shown in Fig. 15.

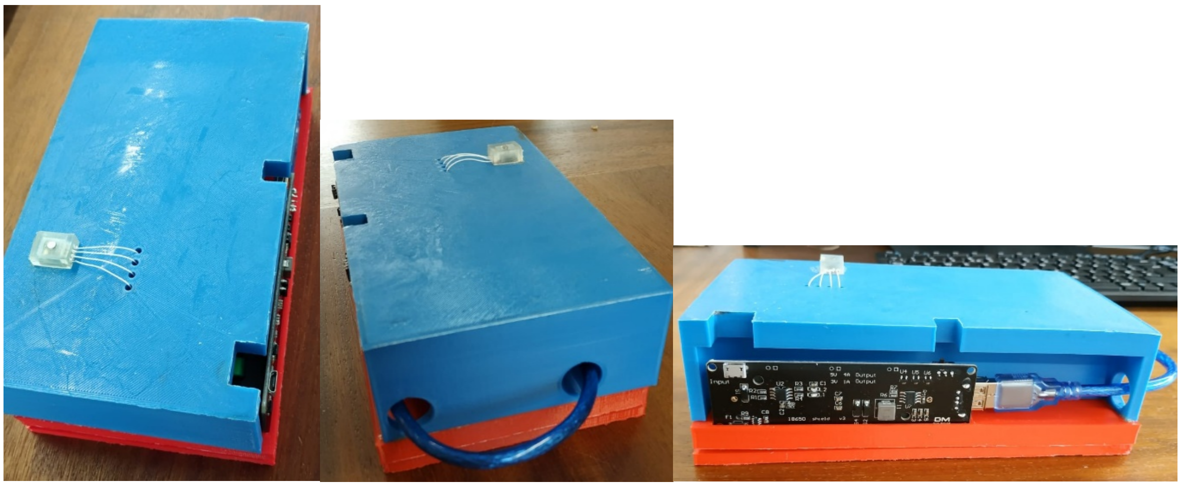


Figure 15: Prototype: Physical device prototype for wireless tactile sensing module (left to right – top, back and side views).

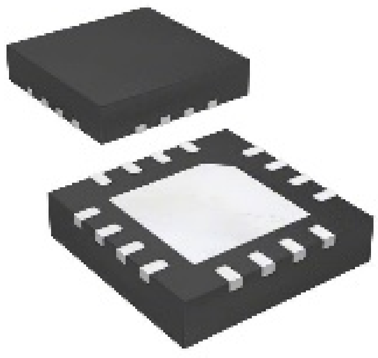
IV. DISCUSSION

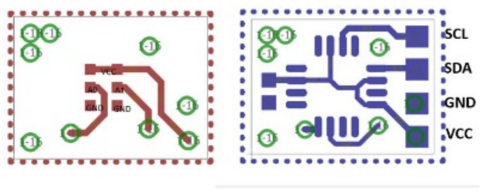
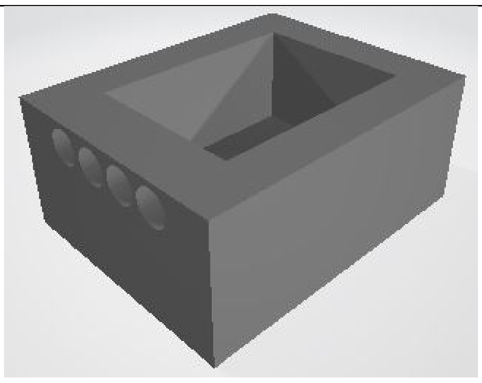
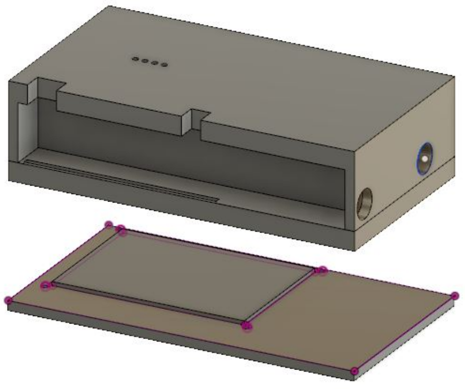
Ideally, the tactile sensing solution should be made as economical and cost-effective as possible, hence we take a closer look at the breakdown of the components. The sensitivity of the soft sensor is a vital aspect to consider, so an experiment to test its sensitivity is conducted. The limitations and potential applications are also discussed.

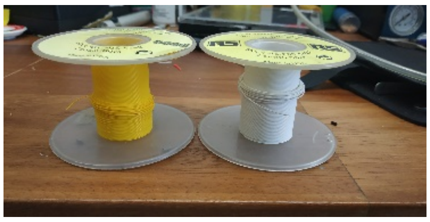
A. Cost Analysis

The cost of all the individual components used in this project are listed in Tab. 3, including their details and images. The components can be divided into two sections, the sensing unit and readout electronics.

Table 3: Cost of components used

#	Component Details	Product ID	Cost (£) per unit	Images
Sensing Unit				
1.	Melexis Triaxis® Magnetic Field Sensor Magnetometer	MLX90393	1.49	

2.	N42 Neodymium Magnet - 0.19kg Pull (3mm diameter x 1mm thickness)	F331-50	0.11	
3.	Printed Circuit Board	Custom	50	
4.	Form Labs Elastic Resin 3D Print Shape (0.61 mL)	RS-F2-ELCL-01	0.098	
5.	Ultimaker PLA 3D Print Shape Blue and Red (176g body, 59g head, 66g leg)	RAL 5002; RAL 3020	15.41	
Readout Electronics				

6.	Espressif ESP32 DevKit V1 Microcontroller (WiFi and BLE)	ESP-WROOM-32	7.00	
7.	DIY More ESP32 18650 Battery Shield V3	SKU:012590	6.50	
8.	Samsung 25R 18650 Battery	SKU: ES18650-SAMS-25R	3.79	
9.	Kynar Insulated 30awg (0.25mm) ultra-thin Soldering Wires	209-4798	0.40	

10.	Resistors (4.3kΩ) (2 pcs)	CFR-25JT- 52-20K	0.18	
11.	Stripboard	Custom	2.00	
Total Cost			**£86.978**	

The total cost of acquiring and assembling all the parts together to produce the wireless tactile sensing device adds up to approximately £87. This number may seem like a lot but it can be significantly reduced. For example, the custom-made PCB which costs £50 per unit can be brought down to as low as £5 per unit, if mass production was done. Additionally, the cost of materials (£15.41) used to manufacture the housing platform, which is mainly done for presentation purposes, can also be reduced to nothing for a specific application that may not need this. Taking these possibilities of cost reductions into account, the total cost to produce such a wireless tactile sensing module could be as low as £26.60.

B. Sensitivity Experiment

Magnetic Hall-effect sensors measure the changes in magnetic flux which are proportional to the forces applied on the tactile sensor. A previous study using the same sensing chip and similar soft material sensor body measured its sensitivity with the minimum detected force being about 0.007N [52].

The sensitivity experiment for this project is conducted by taking 10 readings of each axis arbitrarily when no external force is applied on the sensor, and then taking a further 10 readings of each axis arbitrarily when a soft finger press force is exerted on the sensor. Taking the mean values of the different readings in the three axes, it can be determined how sensitive the sensor is to a soft touch force.

There is a natural magnetic field existent on magnetic objects, and the flux values constantly change as shown in Tab. 4, even if the changes are small. When no external force is applied, the MLX90393 sensor provides the readings of the magnetic flux in X, Y and Z axes generated by the proximity of the small magnet on top of it. These readings keep fluctuating, but stay within similar ranges. Therefore, the mean values of 4790, 2149 and 22689 correspond to the magnetic flux readings of X, Y and Z axes respectively, as shown in Tab. 4:

Table 4: Sensor readings when no force applied											**Mean**
X	4794	4797	4786	4787	4790	4788	4789	4791	4784	4792	**4790**
Y	2150	2149	2155	2148	2146	2152	2151	2146	2144	2148	**2149**
Z	22694	22689	22687	22682	22686	22684	22694	22696	22690	22688	**22689**

When an external force is applied, the magnetic flux readings are expected to change significantly, because the magnet within the soft shape changes its position and gets closer to the MLX90393 sensor. This means that there will be stronger magnetic field near the sensor, resultantly producing higher sensor readings. Tab. 5 shows 10 arbitrary sensor readings for each axis when a soft index finger press is applied onto the sensor, deforming the soft shape and moving the magnet. The mean values of 6012, 10591 and 45770 correspond to the magnetic flux readings in the X, Y and Z axes respectively, as shown in Tab. 5:

Table 5: Sensor readings when finger press applied											**Mean**
X	5455	5241	5662	6063	6375	6415	7640	6019	5843	5411	**6012**
Y	11193	11436	11085	10930	10606	9770	9849	10204	10487	10348	**10591**
Z	45744	45768	45752	45761	45780	45790	45793	45774	45766	45772	**45770**

When comparing the differences in mean readings upon exerting an external soft force, there is a change of 1222 in the X-axis, 8442 in the Y-axis and 23081 in the Z-axis. In terms of percentage changes, the magnetic field changes by 25.5% in the X-axis, 392.8% in the Y-axis and 101.7% in the Z-axis. The most significant change is observed in the Y-axis because of the vertical finger press producing normal force, thus changing the vertical Y-axis magnetic flux reading most dramatically. There are less changes of magnetic field readings in X and Z axes, because of minimum movements of the magnet in those directions.

C. Limitations & Challenges

The wireless tactile sensing solution implemented in this project was based on commercially available components. Although the individual components selected for this project are state-of-the-art in the current market, they do impose certain limitations as well.

To implement the wireless data transfer capabilities, a microcontroller and its power source are required, which add to the overall size and cost of the tactile sensing module. The microcontroller may face challenges to connect to a nearby Wi-Fi device, as it sometimes stalls while searching for the device even if it is well within range. This may be solved by manufacturing a custom PCB with the sensing chip as well as a Wi-Fi chip, that can still be compact in size and eliminate the requirement of a microcontroller. If the sensing module is integrated in a device that already contains a processing unit with wireless communication capabilities, then they can be paired together without further a microcontroller. The rechargeable battery used as the power source is bulky and needs to be regularly charged for consistent operation. Since this technology has low power requirements, it may be possible to use renewable energy resources such as solar panels. However, this needs to be further explored in terms of energy efficiency, size and cost.

In terms of the sensing component, the soft shape characteristically introduces hysteresis. Hysteresis is the delay in time for a soft material to return to its original state and can be measured by the following formula:

$$Hysteresis\ \% = \frac{(F_{mu} - F_{ml})}{(F_{max} - F_{min})}\ X\ 100\%$$

F_{min} and F_{max} are the minimum and maximum average forces measured by reference sensor respectively; whereas, F_{ml} and F_{mu} are the values of loading and unloading cycles respectively [34]. When reading digital signals, the Signal-to-Noise Ratio (SNR) should also be considered. SNR is measured by the following formula:

$$SNR_{dB} = 20log_{10}\left(\frac{\mu_u - \mu_p}{\sigma_u}\right)dB$$

μ_u is the mean and σ_u is the standard deviation of the measurement when no force is applied; whereas, μ_p is the mean when force is applied [34]. Besides these, the durability and sensitivity of the soft sensor should also be explored before considering its applications.

Before building the prototype, there were many challenges along the way. In order to 3D print the shapes, learning CAD designing skills was crucial. While testing the circuit, a small magnet accidentally got attached to a soldered part of the sensing PCB, hence shorting the circuit and instantly burning out the ESP32. It was also initially a challenge to obtain readings because of not knowing the correct address of the sensor. While designing the housing platform, the measurements required precision, and failure to incorporate one small detail resulted in the whole 3D print to get discarded. Regardless to say, these challenges imposed constraints in time and completion of the project. However, all the aforementioned challenges were successfully overcome and the project was successfully completed.

D. Applications

The implemented wireless tactile sensing solution can measure forces in three axes using a sensor that is soft, low-cost and small in size. There is a lot of research being done into efficient tactile sensing because of the progressing fields of robotics and Internet of Things. Many researchers estimate a huge potential for the blossoming sensor and data collection industry. In fact, there were 7 billion connected devices in 2018 and 10 billion devices expected to be connected by 2020, and these numbers do not include smartphones, telephones, tablets or laptops [70]. This clearly suggests the colossal demand of Internet of Things devices, which largely comprise of sensors.

Tactile skin can be used in robots to revolutionise service industry including human prosthetics and augmented reality using smart textiles [1]. In a previous work, a wireless modular sensor with a transceiver was developed that can be mounted on a shoe for gait activity analysis [71]. There can be several different applications of wireless tactile sensing modules such as measuring pressure in a gas cylinder, triggering security alarm, sensing in human prostheses, mini joystick button and so on. It can be utilised in sports such as a smart football boot to train young players improve shots on their weaker foot. It can also be used for injury rehabilitation where the movements of the patient are monitored and minimised. It can have applications in virtual reality sensing devices such as controllers. It can be used in robot fingers because of its small size and good sensitivity, and successfully perform complex tasks such as medical surgeries. Hence, a wireless tactile sensor can be very useful in a wide range of remote applications with proper data visualisation tools.

V. CONCLUSION

Tactile sensors can collect data from the world and have a wide range of applications in multitudes of industries. The tactile sensor used in this project is a Hall-effect magnetometer that can measure normal and shear forces in three axes. Even at a soft applied touch force, the soft tactile sensor module has high sensitivity and precision. Having wireless capabilities of this sensor eliminate the usage of bulky wiring and possibility of easy integration in remote applications. In this project, the wireless solution presented was based on electronic components currently commercially available. However, there are limitations of this solution mainly relating to its size. To enable wireless data collection and visualisation, an ESP32 microcontroller is used, which is among the most compact microcontrollers out there, but still adds to the overall size of the tactile sensing unit. Furthermore, the battery holder used to power the ESP32 is also quite bulky compared to the sensor itself. These two components mainly add to the size of the solution. In the future, custom-made electronics can enable successful miniaturisation of this wireless solution. If an integrated chip was mounted onto the sensor PCB for wireless connectivity similar to the one used on the ESP32, then the usage of a microcontroller can possibly be avoided. Alternatively, a wireless transceiver could be considered. Smaller batteries can be used to power the sensing module, if a microcontroller is not used. Although these upgrades may impose challenges and be costly, they could provide a truly efficient, effective and miniature wireless tactile sensing module.

VI. BIBLIOGRAPHY

[1] R. S. Dahiya, P. Mittendorfer, M. Valle, G. Cheng, and V. J. Lumelsky, *"Directions toward effective utilization of tactile skin: A review,"* IEEE Sensors Journal, vol. 13, no. 11, 2013, pp. 4121–4138.

[2] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, *"Tactile sensing–from humans to humanoids,"* IEEE Transactions on Robotics, vol. 26, no. 1, 2010, pp. 1–20.

[3] Z. Li, P. Hsu, and S. Sastry, *"Grasping and coordinated manipulation by a multifingered robot hand,"* Int. J. Robot. Res., vol. 8, no. 4, Aug. 1989, pp. 33–50.

[4] R. D. Howe and M. R. Cutkosky, *"Integrating tactile sensing with control for dextrous manipulation,"* in Proc. IEEE Int. Workshop Intell. Motion Control, Istanbul, Turkey, Aug. 1990, vol. 1, pp. 369–374.

[5] A. D. Berger and P. K. Khosla, *"Using tactile data for real-time feedback,"* Int. J. Robot. Res., vol. 10, no. 2, 1991, pp. 88–102.

[6] R. S. Fearing, *"Tactile sensing mechanisms,"* Int. J. Robot. Res., vol. 9, no. 3, 1990, pp. 3–23.

[7] K. Weiss and H. Worn, *"Tactile sensor system for an anthropomorphic robotic hand,"* in Proc. IEEE Int. Conf. Manipulation Grasping, Genoa, Italy, 2004, pp. 1–6.

[8] D. J. Beebe, A. S. Hsieh, D. D. Denton, and R. G. Radwin, *"A silicon force sensor for robotics and medicine,"* Sens. Actuators A, vol. 50, 1995, pp. 55–65.

[9] M. R. Wolffenbuttel and P. P. L. Regtien, *"Polysilicon bridges for the realization of tactile sensors,"* Sens. Actuators A, vol. A26, no. 1–3, 1991, pp. 257–264.

[10] D. Bloor, K. Donnelly, P. J. Hands, P. Laughlin, and D. Lussey, *"A metalpolymer composite with unusual properties,"* J. Phys. D: Appl. Phys., vol. 38, no. 16, 2005, pp. 2851–2860.

[11] R. Walker, *"Developments in dextrous hands for advanced robotic applications,"* presented at the 10th Int. Symp. Robot. App., Spain, 2004.

[12] V. Maheshwari and R. F. Saraf, *"High-resolution thin-film device to sense texture by touch,"* Science, vol. 312, 2006, pp. 1501–1504.

[13] B. L. Gray and R. S. Fearing, *"A surface micromachined microtactile sensor array,"* in Proc. Int. Conf. Robot. Automat., Minneapolis, MN, vol. 1, 1996, pp. 1–6.

[14] P. A. Schmidt, E. Mael, and R. P. Wurtz, *"A sensor for dynamic tactile information with applications in human-robot interaction & object exploration,"* Robot. Autonomous Syst., vol. 54, 2006, pp. 1005–1014.

[15] M. Maggiali, G. Cannata, P. Maiolino, G. Metta, M. Randazzao, and G. Sandini, *"Embedded tactile sensor modules,"* presented at the 11th Mechatron. Forum Biennial Int. Conf., Limerick, Ireland, 2008.

[16] J.-S. Heo, J.-H. Chung, and J.-J. Lee, *"Tactile sensor arrays using fiber bragg grating,"* Sens. Actuators A, vol. 126, 2006, pp. 312–327.

[17] M. Ohka, H. Kobayashi, J. Takata, and Y. Mitsuya, *"Sensing precision of an optical three-axis tactile sensor for a robotic finger,"* in Proc. 15th Int. Symp. Robot Human Interactive Commun. "RO-MAN," Hatfield, U.K., 2006, pp. 214–219.

[18] S. Ando and H. Shinoda, *"Ultrasonic emission tactile sensing,"* IEEE Control Syst. Mag., vol. 15, no. 1, Feb. 1995, pp. 61–69.

[19] J. R. Flanagan and A. M. Wing, *“Modulation of grip force with load force during point-to-point arm movements,”* Exp. Brain Res., vol. 95, 1993, pp. 131–143.

[20] E. Torres-Jara, I. Vasilescu, and R. Coral, *“A soft touch: Compliant tactile sensors for sensitive manipulation,”* CSAIL, Mass. Inst. Technol., Cambridge, MA, Tech. Rep. MIT-CSAIL-TR-2006-014, 2006.

[21] L. Jamone, G. Metta, F. Nori, and G. Sandini, *“James: A humanoid robot acting over an unstructured world,”* in Proc. 6th IEEE-RAS Int. Conf. Humanoid Robots, Genoa, Italy, 2006, pp. 143–150.

[22] T. J. Nelson, R. B. V. Dover, S. Jin, S. Hackwood, and G. Beni, *“Shearsensitive magnetoresistive robotic tactile sensor,”* IEEE Trans. Magn., vol. MAG-22, no. 5, Mar. 1986, pp. 394–396.

[23] J. J. Clark, *“A magnetic field based compliance matching sensor for high resolution, high compliance tactile sensing,”* in ICRA. IEEE, 1988, pp. 772–777.

[24] W. C. Nowlin, *“Experimental results on Bayesian algorithms for interpreting compliant tactile sensing data,”* in ICRA. IEEE, 1991, pp. 378–383.

[25] T. P. Tomo, S. Somlor, A. Schmitz, L. Jamone, W. Huang, H. Kristanto, and S. Sugano, *“Design and characterization of a three-axis hall effectbased soft skin sensor,”* Sensors, vol. 16, no. 4, 2016, p. 491.

[26] T. P. Tomo, W. K. Wong, A. Schmitz, H. Kristanto, A. Sarazin, L. Jamone, S. Somlor, and S. Sugano, *“A modular, distributed, soft, 3-axis sensor system for robot hands,”* in 2016 IEEE-RAS Humanoids. IEEE, 2016.

[27] T. P. Tomo, A. Schmitz, W. K. Wong, H. Kristanto, S. Somlor, J. Hwang, L. Jamone, and S. Sugano, *“Covering a robot fingertip with uSkin: A soft electronic skin with distributed 3-axis force sensitive elements for robot hands,”* in IEEE/RSJ IROS. IEEE, 2017.

[28] L. Jamone, L. Natale, G. Metta, and G. Sandini, *“Highly sensitive soft tactile sensors for an anthropomorphic robotic hand,”* IEEE sensors Journal, vol. 15, no. 8, 2015, pp. 4226–4233.

[29] C. Ledermann, S. Wirges, D. Oertel, M. Mende, and H. Woern, *“Tactile sensor on a magnetic basis using novel 3D Hall Sensor - First prototypes and results,”* in 17th International Conference on Intelligent Engineering Systems. IEEE, 2013, pp. 55–60.

[30] T. Yoshikai, M. Hayashi, Y. Ishizaka, H. Fukushima, A. Kadowaki, T. Sagisaka, K. Kobayashi, I. Kumagai, and M. Inaba, *“Development of robots with soft sensor flesh for achieving close interaction behavior,”* Advances in Artificial Intelligence, vol. 2012, p. 8.

[31] H. Iwata and S. Sugano, *“Design of human symbiotic robot TWENDYONE,”* in ICRA. IEEE, 2009, pp. 580–586.

[32] A. Schmitz, P. Maiolino, M. Maggiali, L. Natale, G. Cannata, and G. Metta, *“Methods and technologies for the implementation of largescale robot tactile sensors,”* IEEE Transactions on Robotics, vol. 27, no. 3, 2011, pp. 389–400.

[33] S. Youssefian, N. Rahbar, and E. Torres-Jara, *“Contact behavior of soft spherical tactile sensors,”* IEEE Sensors Journal, vol. 14, no. 5, 2014, pp. 1435– 1442.

[34] T. P. Tomo, M. Regoli, A. Schmitz, L. Natale, H. Kristanto, S. Somlor, L. Jamone, G. Metta and S. Sugano, *“A New Silicone Structure for uSkin - a Soft, Distributed, Digital 3-axis Skin Sensor - and its Integration on the Humanoid Robot iCub,”* IEEE Robotics and Automation Letters, vol. 3, no. 3, July 2018, pp. 2584-2591.

[35] H. Shinoda and H. Oasa, *"Wireless tactile sensing element using stress-sensitive resonator,"* in IEEE/ASME Transactions on Mechatronics, vol. 5, no. 3, Sept. 2000, pp. 258-265.

[36] I. Peter, D. J. Holding, K. J. Blow, B. Tam, Xianghong Ma and P. N. Brett, *"The design of a flexible digit towards wireless tactile sense feedback,"* ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004., Kunming, China, 2004, pp. 468-473 Vol. 1.

[37] D. K. Pai and P. Rizun, *"The WHaT: a wireless haptic texture sensor,"* 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings., Los Angeles, CA, USA, 2003, pp. 3-9.

[38] Y. Makino, A. Kamigori and T. Maeno, *"Wireless wearable vibration sensor for touch-based life log system,"* 2012 Ninth International Conference on Networked Sensing (INSS), Antwerp, 2012, pp. 1-4.

[39] T. McDaniel, S. Krishna, D. Villanueva and S. Panchanathan, *"A haptic belt for vibrotactile communication,"* 2010 IEEE International Symposium on Haptic Audio Visual Environments and Games, Phoenix, AZ, 2010, pp. 1-2.

[40] A. Stan, N. Botezatu and R. G. Lupu, *"The Design of a Scalable Haptic System Used for Impaired People Assistance,"* 2015 20th International Conference on Control Systems and Computer Science, Bucharest, 2015, pp. 863-866.

[41] A. Keyes, M. D'Souza and A. Postula, *"Navigation for the blind using a wireless sensor haptic glove,"* 2015 4th Mediterranean Conference on Embedded Computing (MECO), Budva, 2015, pp. 361-364.

[42] M. Beccani, C. Di Natali, M. E. Rentschler and P. Valdastri, *"Wireless tissue palpation: Proof of concept for a single degree of freedom,"* 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, 2013, pp. 711-717.

[43] P. M. Ros, M. Crepaldi, A. Bonanno and D. Demarchi, *"Wireless Multi-channel Quasi-digital Tactile Sensing Glove-Based System,"* 2013 Euromicro Conference on Digital System Design, Los Alamitos, CA, 2013, pp. 673-680.

[44] S. Yeh, H. Chang and W. Fang, *"A novel polymer filled CMOS-MEMS inductive-type tactile sensor with wireless sensing capability,"* 2017 19th International Conference on Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS), Kaohsiung, 2017, pp. 834-837.

[45] C. Rojas and J. Decotignie, *"Artificial skin for human prostheses, enabled through wireless sensor networks,"* 2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Hsinchu, 2017, pp. 1-8.

[46] T. Sun, T. Tsukamoto, T. Ishikawa and S. Tanaka, *"System development of biosensing module using CMOS hall sensor array for disposable wireless diagnosis device,"* 2017 IEEE 12th International Conference on Nano/Micro Engineered and Molecular Systems (NEMS), Los Angeles, CA, 2017, pp. 160-163.

[47] L. G. Ni, D. P. Kari, A. Muganza, B. Dushime and A. N. Zebaze, *"Wireless integration of tactile sensing on the hand of a humanoid robot NAO,"* 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, Paris, 2012, pp. 982-988.

[48] J. H. Won, G. H. Yang and J. W. Jeon, *"A novel wireless vibrotactile display device for representing 3DOF force feedback in teleoperation,"* 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Busan, 2015, pp. 1749-1753.

[49] T. Hongell, I. Kivelä and I. Hakala, *"Wireless strain gauge network — Best-hall measurement case,"* 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 2014, pp. 1-6.

[50] K. Kyung, J. Lee and M. A. Srinivasan, *"Precise manipulation of GUI on a touch screen with haptic cues,"* World Haptics 2009 - Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Salt Lake City, UT, 2009, pp. 202-207.

[51] Melexis – "*Triaxis® micropower magnetometer (magnetic field sensor IC) MLX90393 product datasheet*," Available from: https://www.melexis.com/en/product/mlx90393/triaxismicropower-magnetometer. [Accessed 17th August 2019].

[52] T. Paulino, P. Ribeiro, M. Neto, S. Cardoso, A. Schmitz, J. Santos-Victor, A. Bernardino, and L. Jamone, *"Low-cost 3-axis soft tactile sensors for the human-friendly robot Vizzy,"* in ICRA, 2017.

[53] Formlabs Inc. – *"3D Printing Materials for Engineering, Manufacturing, and Product Design,"* Available from: https://formlabs.com/materials/engineering/ [Accessed 17th August, 2019].

[54] Janakiram MSV – "*10 DIY Development Boards for IoT Prototyping*," Available from: https://thenewstack.io/10-diy-development-boards-iot-prototyping/ [Accessed 17th August, 2019].

[55] Sara Santos – *"ESP32 vs ESP8266 – Pros and Cons,"* Available from: https://makeradvisor.com/esp32-vs-esp8266/ [Accessed 17th August, 2019].

[56] Espressif Systems – “*ESP32 Series Datasheet,”* Available from: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf [Accessed 17th August, 2019].

[57] Espressif Systems – “*ESP32 A Different IoT Power and Performance,”* Available from: https://www.espressif.com/en/products/hardware/esp32/overview [Accessed 17th August, 2019].

[58] I2C Info – *“I2C Info – I2C Bus, Interface and Protocol,”* Available from: https://i2c.info/ [Accessed 17th August, 2019].

[59] Espressif Systems – “*ESP32 Technical Reference Manual,”* Available from: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf [Accessed 17th August, 2019].

[60] Vishnu M Aiea – *“A Better Pinout Diagram for ESP32 DEVKIT Development Board,”* Available from: https://www.reddit.com/r/esp32/comments/b5mkag/a_better_pinout_diagram_for_esp32_devkit/ [Accessed 17th August, 2019].

[61] RadioShuttle Network Protocol – *“Battery-Powered ESP32,* “Available from: https://www.radioshuttle.de/en/media-en/tech-infos-en/battery-powered-esp32/ [Accessed 17th August, 2019].

[62] 18650.uk – *“Samsung 25R 18650 Battery,”* Available from: https://18650.uk/product/samsung-25r-18650-battery/ [Accessed 17th August, 2019].

[63] DIY More.cc – *“Micro USB Wemos ESP32 18650 Battery Shield V3 ESP-32 For Arduino Raspberry Pi,”* Available from: https://www.diymore.cc/products/micro-usb-

wemos-esp32-18650-battery-shield-v3-esp-32-for-arduino-raspberry-pi [Accessed 17th August, 2019].

[64] Ultimaker – “*Ultimaker 3 series,*” Available from: https://ultimaker.com/3d-printers/ultimaker-3 [Accessed 17th August, 2019].

[65] Ultimaker – “*Ultimaker PLA,*” Available from: https://ultimaker.com/materials/pla [Accessed 17th August 2019].

[66] Formlabs – “*Formlabs Stereolithography,*” Available from: https://formlabs.com/3d-printers/form-3/tech-specs/ [Accessed 17th August, 2019].

[67] Github – “*Async Web Server for ESP8266 and ESP32,*” Available from: https://github.com/me-no-dev/ESPAsyncWebServer [Accessed 17th August, 2019]

[68] Highcharts – “*Highcharts, Highstock and Highmaps documentation,*” Available from: https://www.highcharts.com/docs/ [Accessed 17th August, 2019]

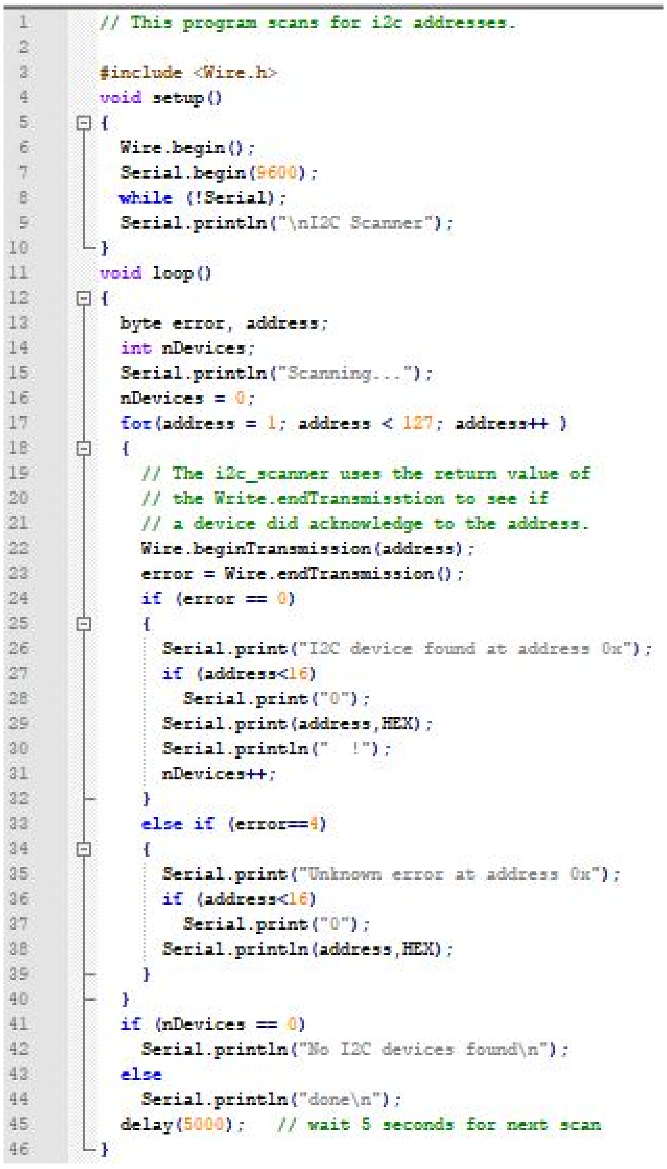
[69] Github – “*Async TCP Library for ESP32*,” Available from: https://github.com/me-no-dev/AsyncTCP [Accessed 17th August, 2019]

[70] Knud Lasse Lueth – “*State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating,*” Aug. 2018, Available from: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/ [Accessed 17th August, 2019]

[71] A. Y. Benbasat, S. J. Morris and J. A. Paradiso, "*A wireless modular sensor architecture and its application in on-shoe gait analysis*," SENSORS, 2003 IEEE, Toronto, Ont., 2003, pp. 1086-1091 Vol.2.

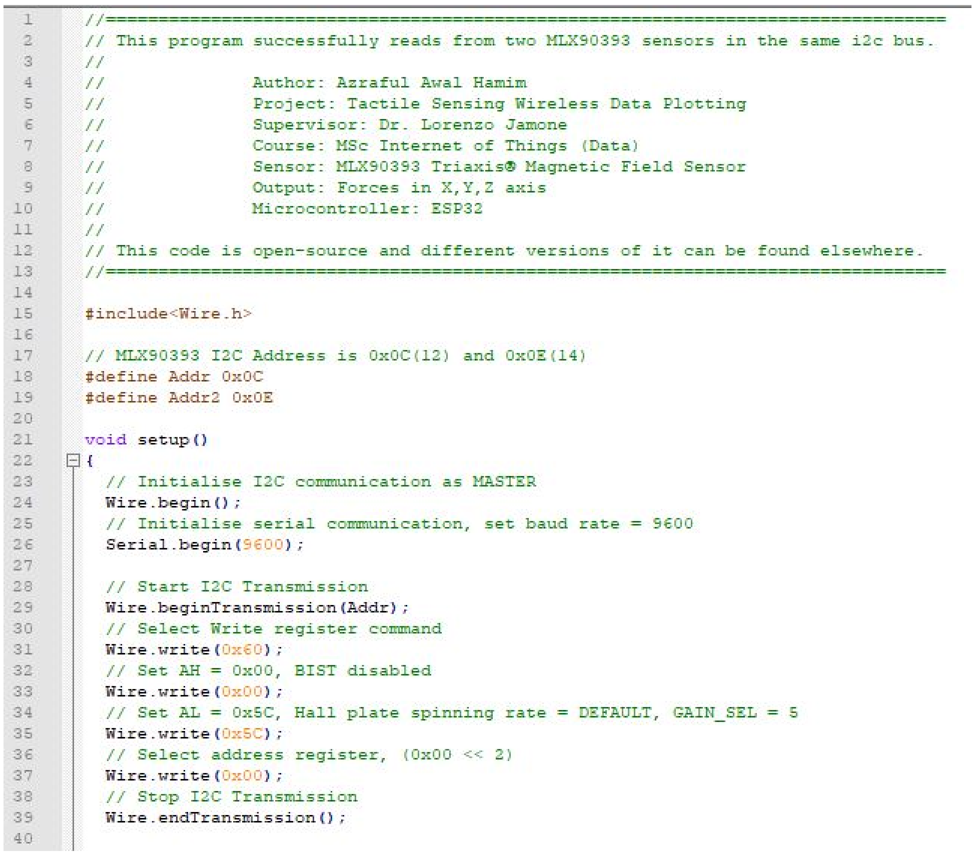
VII. APPENDICES

A. Appendix 1

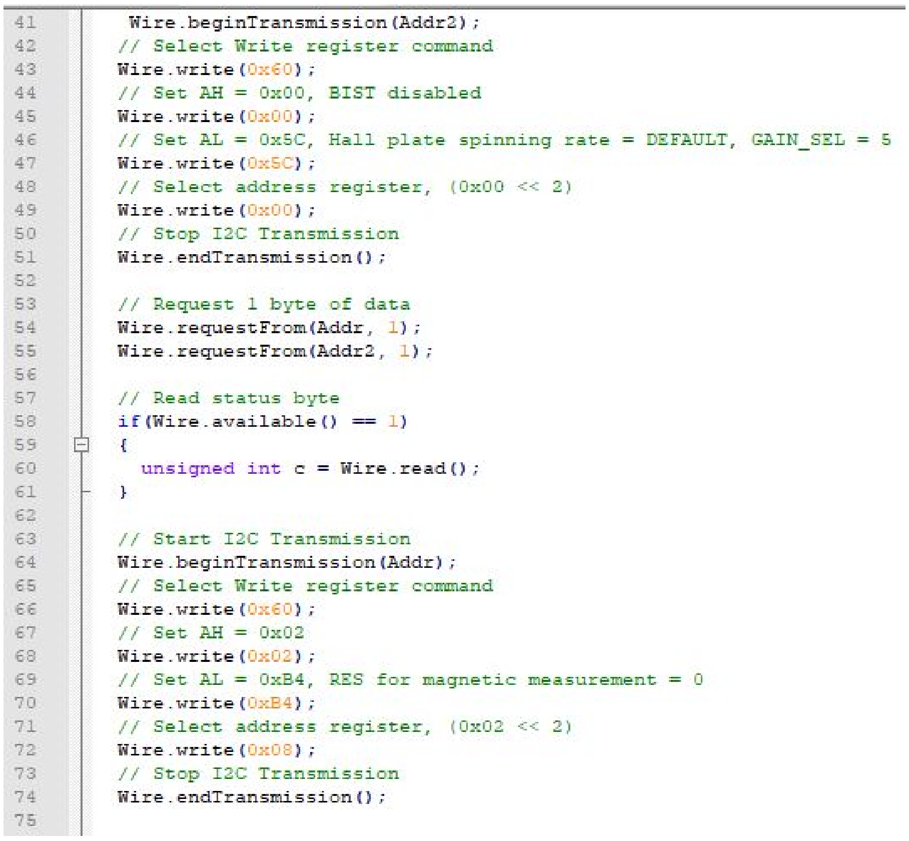


```
// This program scans for i2c addresses.

#include <Wire.h>
void setup()
{
  Wire.begin();
  Serial.begin(9600);
  while (!Serial);
  Serial.println("\nI2C Scanner");
}
void loop()
{
  byte error, address;
  int nDevices;
  Serial.println("Scanning...");
  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");
      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknown error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");
  delay(5000);   // wait 5 seconds for next scan
}
```

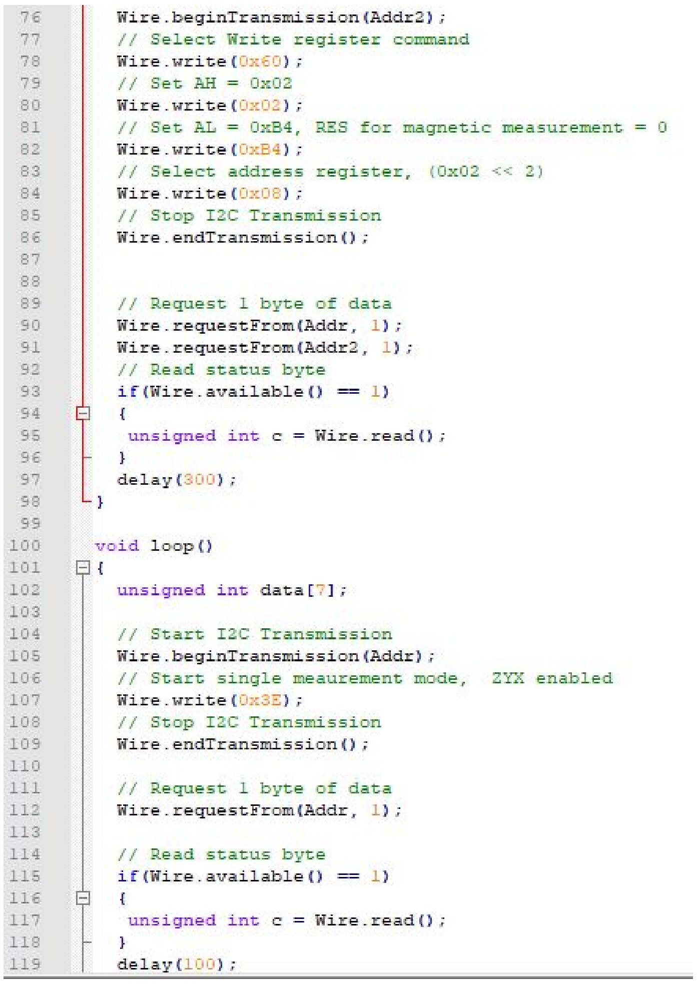
B. Appendix 2



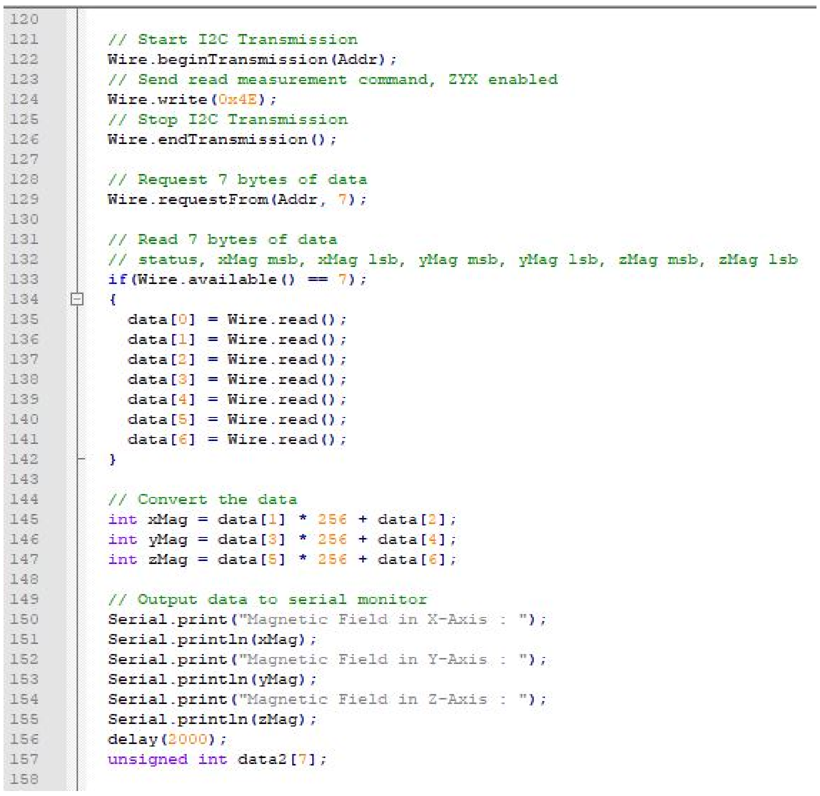
```
//===============================================================================
// This program successfully reads from two MLX90393 sensors in the same i2c bus.
//
//              Author: Azraful Awal Hamim
//              Project: Tactile Sensing Wireless Data Plotting
//              Supervisor: Dr. Lorenzo Jamone
//              Course: MSc Internet of Things (Data)
//              Sensor: MLX90393 Triaxis® Magnetic Field Sensor
//              Output: Forces in X,Y,Z axis
//              Microcontroller: ESP32
//
// This code is open-source and different versions of it can be found elsewhere.
//===============================================================================

#include<Wire.h>

// MLX90393 I2C Address is 0x0C(12) and 0x0E(14)
#define Addr 0x0C
#define Addr2 0x0E

void setup()
{
  // Initialise I2C communication as MASTER
  Wire.begin();
  // Initialise serial communication, set baud rate = 9600
  Serial.begin(9600);

  // Start I2C Transmission
  Wire.beginTransmission(Addr);
  // Select Write register command
  Wire.write(0x60);
  // Set AH = 0x00, BIST disabled
  Wire.write(0x00);
  // Set AL = 0x5C, Hall plate spinning rate = DEFAULT, GAIN_SEL = 5
  Wire.write(0x5C);
  // Select address register, (0x00 << 2)
  Wire.write(0x00);
  // Stop I2C Transmission
  Wire.endTransmission();

```



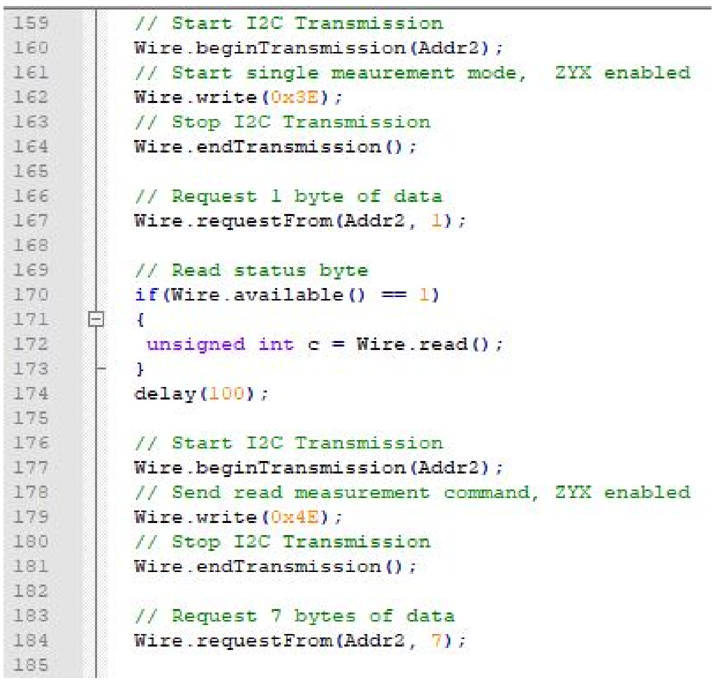
```
   Wire.beginTransmission(Addr2);
  // Select Write register command
  Wire.write(0x60);
  // Set AH = 0x00, BIST disabled
  Wire.write(0x00);
  // Set AL = 0x5C, Hall plate spinning rate = DEFAULT, GAIN_SEL = 5
  Wire.write(0x5C);
  // Select address register, (0x00 << 2)
  Wire.write(0x00);
  // Stop I2C Transmission
  Wire.endTransmission();

  // Request 1 byte of data
  Wire.requestFrom(Addr, 1);
  Wire.requestFrom(Addr2, 1);

  // Read status byte
  if(Wire.available() == 1)
  {
    unsigned int c = Wire.read();
  }

  // Start I2C Transmission
  Wire.beginTransmission(Addr);
  // Select Write register command
  Wire.write(0x60);
  // Set AH = 0x02
  Wire.write(0x02);
  // Set AL = 0xB4, RES for magnetic measurement = 0
  Wire.write(0xB4);
  // Select address register, (0x02 << 2)
  Wire.write(0x08);
  // Stop I2C Transmission
  Wire.endTransmission();

```



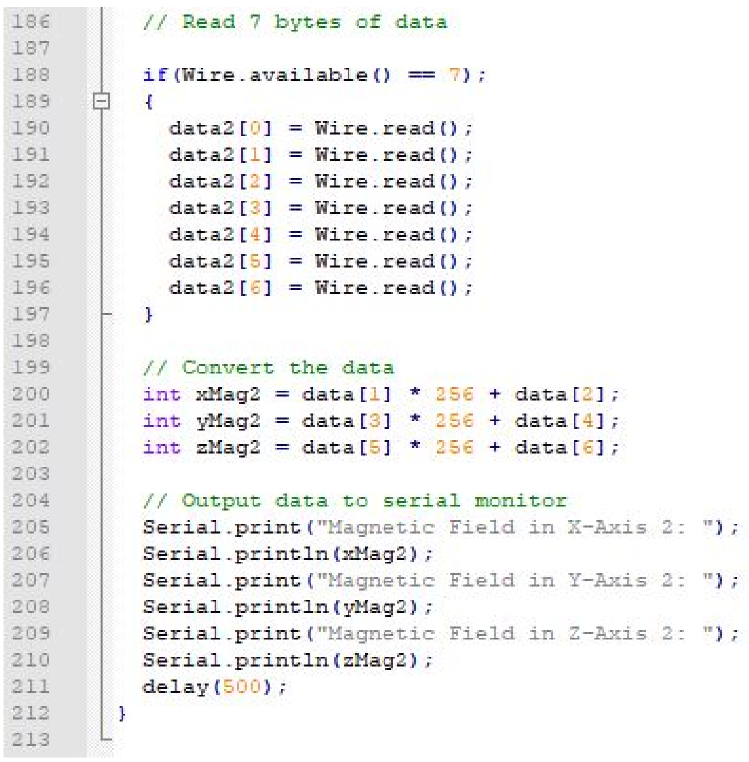
```
  Wire.beginTransmission(Addr2);
  // Select Write register command
  Wire.write(0x60);
  // Set AH = 0x02
  Wire.write(0x02);
  // Set AL = 0xB4, RES for magnetic measurement = 0
  Wire.write(0xB4);
  // Select address register, (0x02 << 2)
  Wire.write(0x08);
  // Stop I2C Transmission
  Wire.endTransmission();


  // Request 1 byte of data
  Wire.requestFrom(Addr, 1);
  Wire.requestFrom(Addr2, 1);
  // Read status byte
  if(Wire.available() == 1)
  {
   unsigned int c = Wire.read();
  }
  delay(300);
}

void loop()
{
  unsigned int data[7];

  // Start I2C Transmission
  Wire.beginTransmission(Addr);
  // Start single meaurement mode,  ZYX enabled
  Wire.write(0x3E);
  // Stop I2C Transmission
  Wire.endTransmission();

  // Request 1 byte of data
  Wire.requestFrom(Addr, 1);

  // Read status byte
  if(Wire.available() == 1)
  {
   unsigned int c = Wire.read();
  }
  delay(100);
```



```

  // Start I2C Transmission
  Wire.beginTransmission(Addr);
  // Send read measurement command, ZYX enabled
  Wire.write(0x4E);
  // Stop I2C Transmission
  Wire.endTransmission();

  // Request 7 bytes of data
  Wire.requestFrom(Addr, 7);

  // Read 7 bytes of data
  // status, xMag msb, xMag lsb, yMag msb, yMag lsb, zMag msb, zMag lsb
  if(Wire.available() == 7);
  {
    data[0] = Wire.read();
    data[1] = Wire.read();
    data[2] = Wire.read();
    data[3] = Wire.read();
    data[4] = Wire.read();
    data[5] = Wire.read();
    data[6] = Wire.read();
  }

  // Convert the data
  int xMag = data[1] * 256 + data[2];
  int yMag = data[3] * 256 + data[4];
  int zMag = data[5] * 256 + data[6];

  // Output data to serial monitor
  Serial.print("Magnetic Field in X-Axis : ");
  Serial.println(xMag);
  Serial.print("Magnetic Field in Y-Axis : ");
  Serial.println(yMag);
  Serial.print("Magnetic Field in Z-Axis : ");
  Serial.println(zMag);
  delay(2000);
  unsigned int data2[7];

```

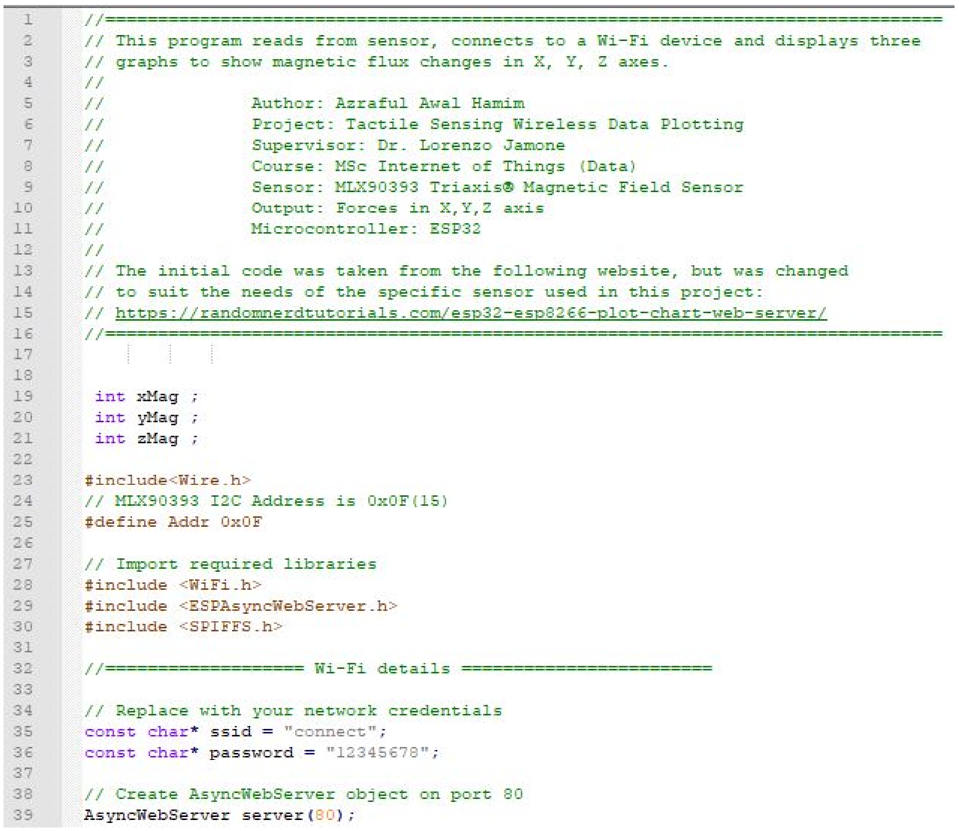


```
 // Start I2C Transmission
 Wire.beginTransmission(Addr2);
 // Start single meaurement mode,  ZYX enabled
 Wire.write(0x3E);
 // Stop I2C Transmission
 Wire.endTransmission();

 // Request 1 byte of data
 Wire.requestFrom(Addr2, 1);

 // Read status byte
 if(Wire.available() == 1)
 {
  unsigned int c = Wire.read();
 }
 delay(100);

 // Start I2C Transmission
 Wire.beginTransmission(Addr2);
 // Send read measurement command, ZYX enabled
 Wire.write(0x4E);
 // Stop I2C Transmission
 Wire.endTransmission();

 // Request 7 bytes of data
 Wire.requestFrom(Addr2, 7);

```

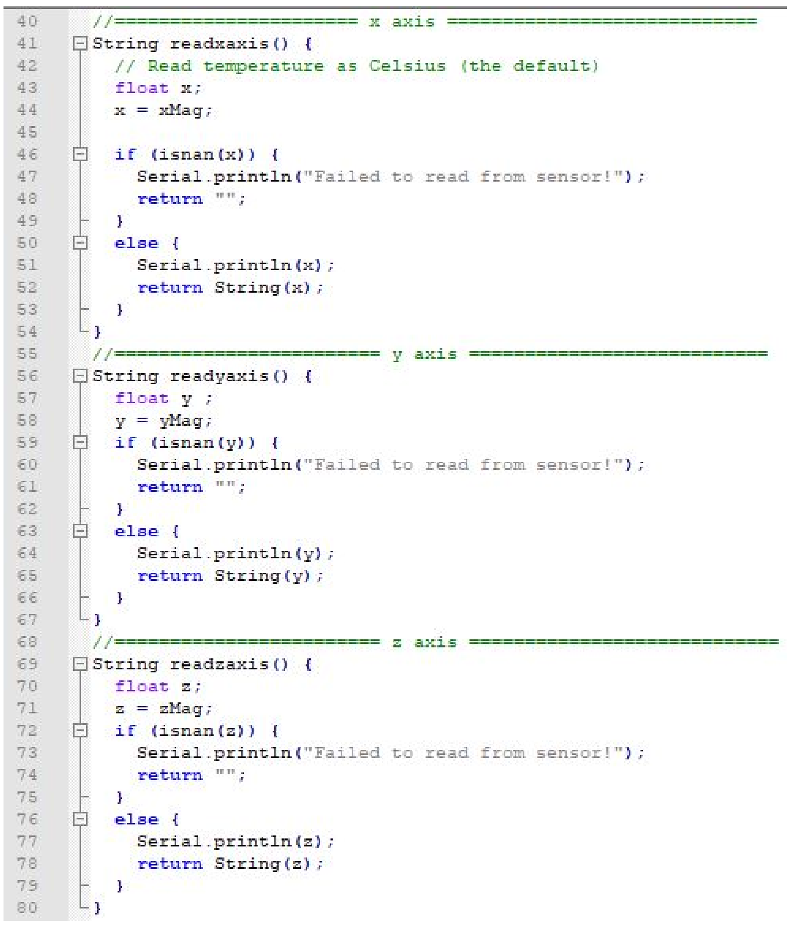


```
  // Read 7 bytes of data

  if(Wire.available() == 7);
  {
    data2[0] = Wire.read();
    data2[1] = Wire.read();
    data2[2] = Wire.read();
    data2[3] = Wire.read();
    data2[4] = Wire.read();
    data2[5] = Wire.read();
    data2[6] = Wire.read();
  }

  // Convert the data
  int xMag2 = data[1] * 256 + data[2];
  int yMag2 = data[3] * 256 + data[4];
  int zMag2 = data[5] * 256 + data[6];

  // Output data to serial monitor
  Serial.print("Magnetic Field in X-Axis 2: ");
  Serial.println(xMag2);
  Serial.print("Magnetic Field in Y-Axis 2: ");
  Serial.println(yMag2);
  Serial.print("Magnetic Field in Z-Axis 2: ");
  Serial.println(zMag2);
  delay(500);
}

```

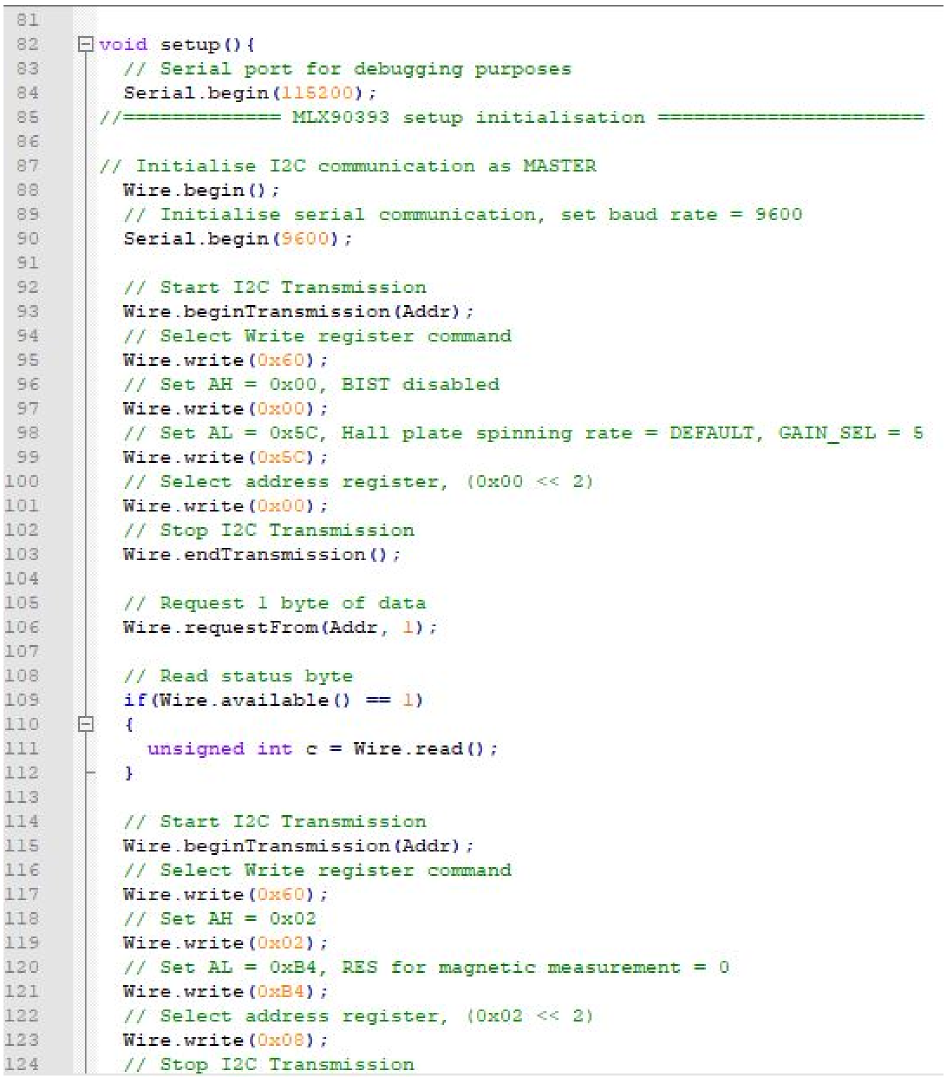
C. Appendix 3



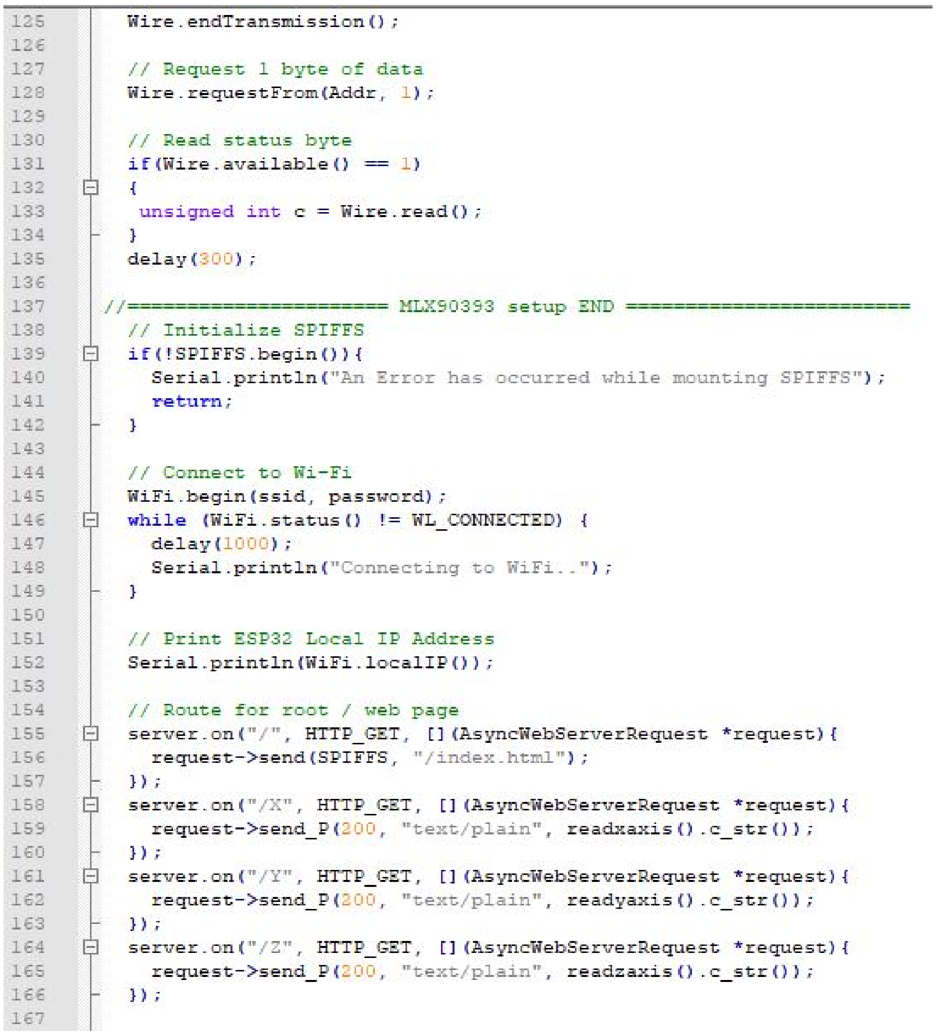
```
//===============================================================================
// This program reads from sensor, connects to a Wi-Fi device and displays three
// graphs to show magnetic flux changes in X, Y, Z axes.
//
//                Author: Azraful Awal Hamim
//                Project: Tactile Sensing Wireless Data Plotting
//                Supervisor: Dr. Lorenzo Jamone
//                Course: MSc Internet of Things (Data)
//                Sensor: MLX90393 Triaxis® Magnetic Field Sensor
//                Output: Forces in X,Y,Z axis
//                Microcontroller: ESP32
//
// The initial code was taken from the following website, but was changed
// to suit the needs of the specific sensor used in this project:
// https://randomnerdtutorials.com/esp32-esp8266-plot-chart-web-server/
//===============================================================================


 int xMag ;
 int yMag ;
 int zMag ;

#include<Wire.h>
// MLX90393 I2C Address is 0x0F(15)
#define Addr 0x0F

// Import required libraries
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>

//=================== Wi-Fi details ========================

// Replace with your network credentials
const char* ssid = "connect";
const char* password = "12345678";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);
```



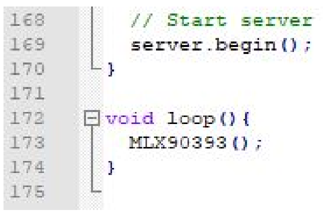
```
 //======================= x axis ============================
String readxaxis() {
  // Read temperature as Celsius (the default)
  float x;
  x = xMag;

  if (isnan(x)) {
    Serial.println("Failed to read from sensor!");
    return "";
  }
  else {
    Serial.println(x);
    return String(x);
  }
}
 //========================= y axis ============================
String readyaxis() {
  float y ;
  y = yMag;
  if (isnan(y)) {
    Serial.println("Failed to read from sensor!");
    return "";
  }
  else {
    Serial.println(y);
    return String(y);
  }
}
 //========================= z axis ============================
String readzaxis() {
  float z;
  z = zMag;
  if (isnan(z)) {
    Serial.println("Failed to read from sensor!");
    return "";
  }
  else {
    Serial.println(z);
    return String(z);
  }
}
```



```
void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);
//============ MLX90393 setup initialisation =====================

// Initialise I2C communication as MASTER
  Wire.begin();
  // Initialise serial communication, set baud rate = 9600
  Serial.begin(9600);

  // Start I2C Transmission
  Wire.beginTransmission(Addr);
  // Select Write register command
  Wire.write(0x60);
  // Set AH = 0x00, BIST disabled
  Wire.write(0x00);
  // Set AL = 0x5C, Hall plate spinning rate = DEFAULT, GAIN_SEL = 5
  Wire.write(0x5C);
  // Select address register, (0x00 << 2)
  Wire.write(0x00);
  // Stop I2C Transmission
  Wire.endTransmission();

  // Request 1 byte of data
  Wire.requestFrom(Addr, 1);

  // Read status byte
  if(Wire.available() == 1)
  {
    unsigned int c = Wire.read();
  }

  // Start I2C Transmission
  Wire.beginTransmission(Addr);
  // Select Write register command
  Wire.write(0x60);
  // Set AH = 0x02
  Wire.write(0x02);
  // Set AL = 0xB4, RES for magnetic measurement = 0
  Wire.write(0xB4);
  // Select address register, (0x02 << 2)
  Wire.write(0x08);
  // Stop I2C Transmission
```

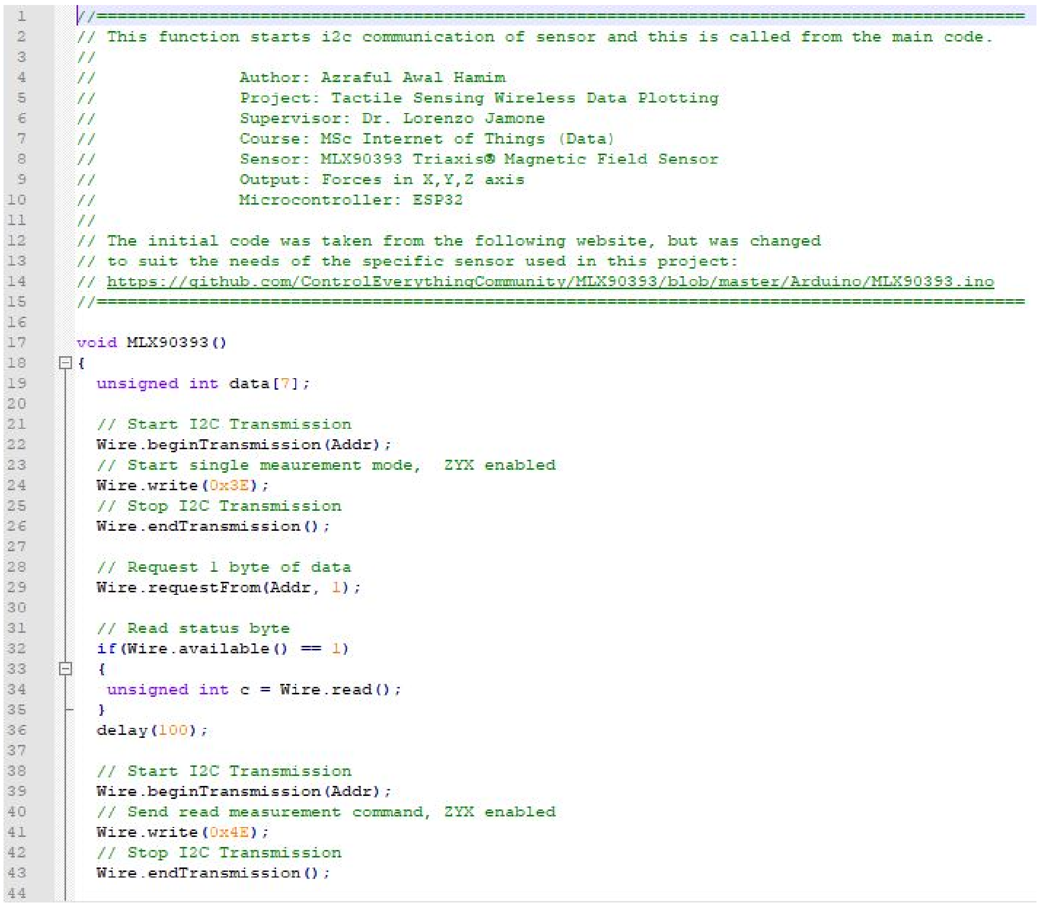


```
  Wire.endTransmission();

  // Request 1 byte of data
  Wire.requestFrom(Addr, 1);

  // Read status byte
  if(Wire.available() == 1)
  {
   unsigned int c = Wire.read();
  }
  delay(300);

//====================== MLX90393 setup END ========================
  // Initialize SPIFFS
  if(!SPIFFS.begin()){
    Serial.println("An Error has occurred while mounting SPIFFS");
    return;
  }

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

  // Print ESP32 Local IP Address
  Serial.println(WiFi.localIP());

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html");
  });
  server.on("/X", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readxaxis().c_str());
  });
  server.on("/Y", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readyaxis().c_str());
  });
  server.on("/Z", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readzaxis().c_str());
  });

```

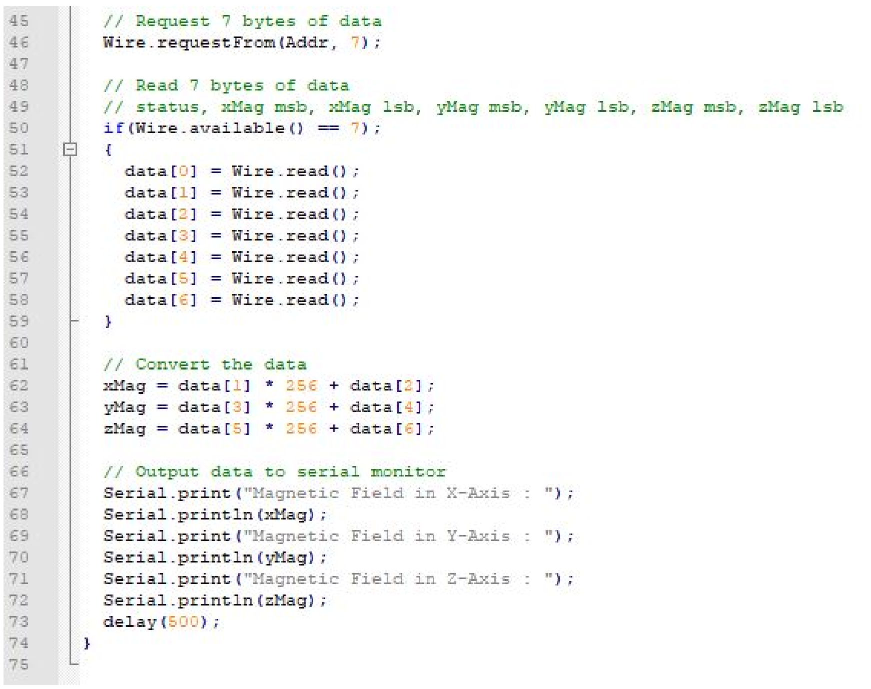


```
  // Start server
  server.begin();
}

void loop(){
  MLX90393();
}

```

D. Appendix 4

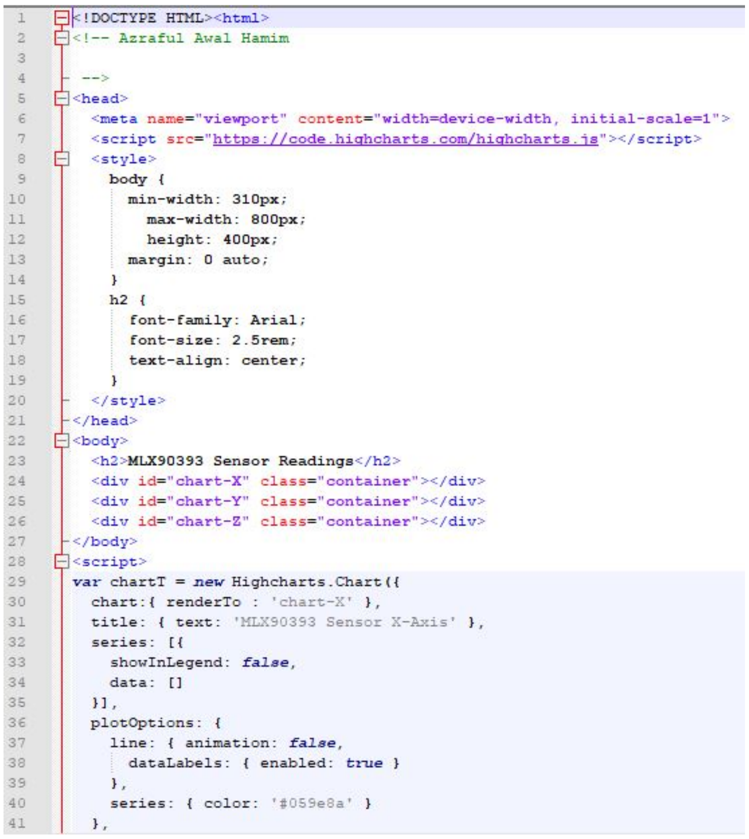


```
//==========================================================================================
// This function starts i2c communication of sensor and this is called from the main code.
//
//                Author: Azraful Awal Hamim
//                Project: Tactile Sensing Wireless Data Plotting
//                Supervisor: Dr. Lorenzo Jamone
//                Course: MSc Internet of Things (Data)
//                Sensor: MLX90393 Triaxis® Magnetic Field Sensor
//                Output: Forces in X,Y,Z axis
//                Microcontroller: ESP32
//
// The initial code was taken from the following website, but was changed
// to suit the needs of the specific sensor used in this project:
// https://github.com/ControlEverythingCommunity/MLX90393/blob/master/Arduino/MLX90393.ino
//==========================================================================================

void MLX90393()
{
  unsigned int data[7];

  // Start I2C Transmission
  Wire.beginTransmission(Addr);
  // Start single meaurement mode,  ZYX enabled
  Wire.write(0x3E);
  // Stop I2C Transmission
  Wire.endTransmission();

  // Request 1 byte of data
  Wire.requestFrom(Addr, 1);

  // Read status byte
  if(Wire.available() == 1)
  {
   unsigned int c = Wire.read();
  }
  delay(100);

  // Start I2C Transmission
  Wire.beginTransmission(Addr);
  // Send read measurement command, ZYX enabled
  Wire.write(0x4E);
  // Stop I2C Transmission
  Wire.endTransmission();
```

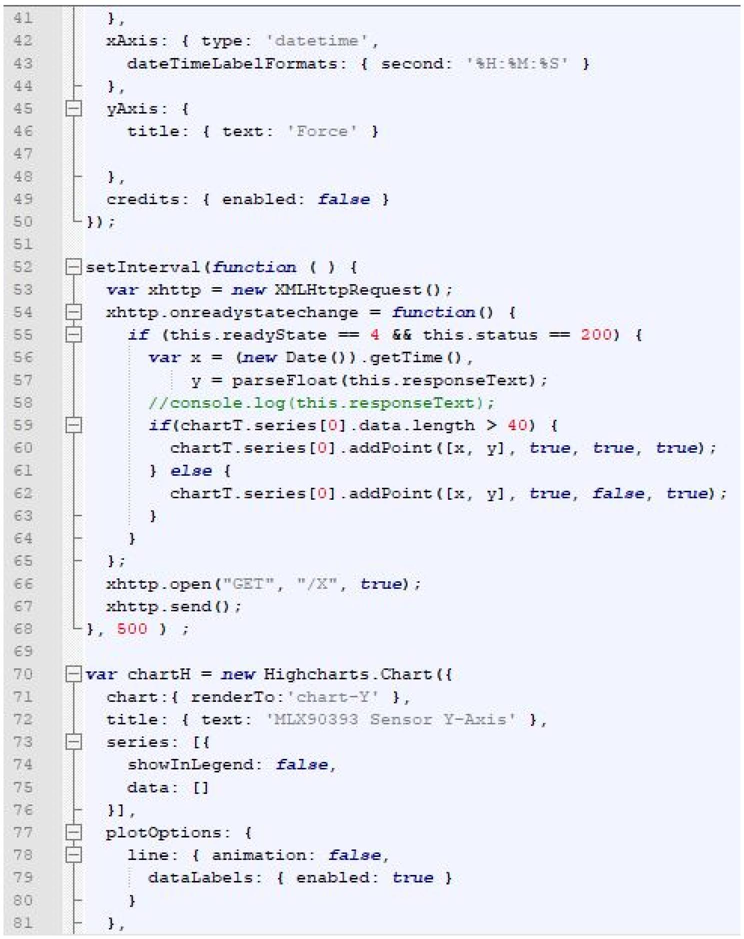


```
  // Request 7 bytes of data
  Wire.requestFrom(Addr, 7);

  // Read 7 bytes of data
  // status, xMag msb, xMag lsb, yMag msb, yMag lsb, zMag msb, zMag lsb
  if(Wire.available() == 7);
  {
    data[0] = Wire.read();
    data[1] = Wire.read();
    data[2] = Wire.read();
    data[3] = Wire.read();
    data[4] = Wire.read();
    data[5] = Wire.read();
    data[6] = Wire.read();
  }

  // Convert the data
  xMag = data[1] * 256 + data[2];
  yMag = data[3] * 256 + data[4];
  zMag = data[5] * 256 + data[6];

  // Output data to serial monitor
  Serial.print("Magnetic Field in X-Axis : ");
  Serial.println(xMag);
  Serial.print("Magnetic Field in Y-Axis : ");
  Serial.println(yMag);
  Serial.print("Magnetic Field in Z-Axis : ");
  Serial.println(zMag);
  delay(500);
}
```

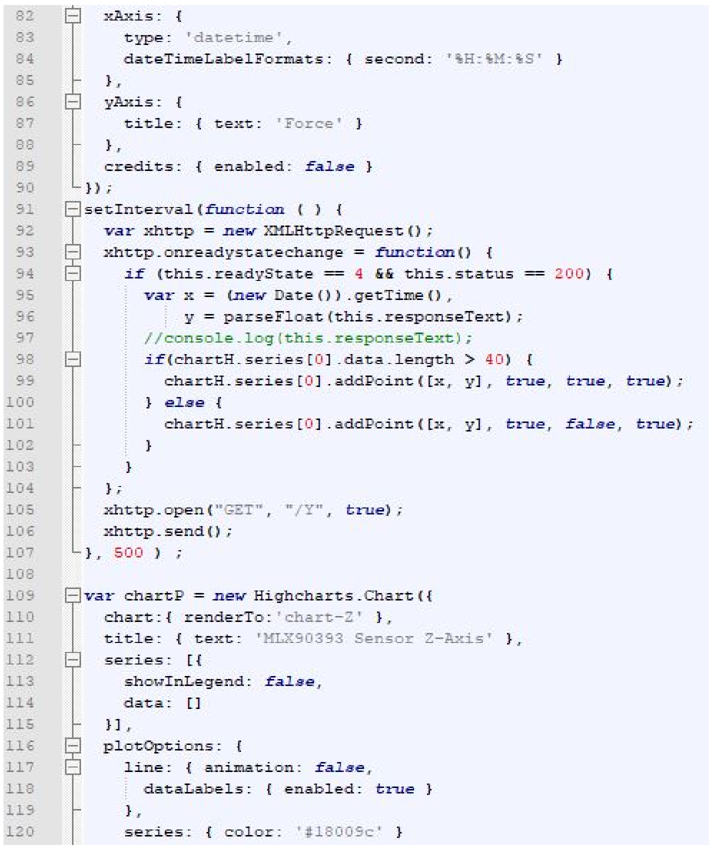
E. Appendix 5:



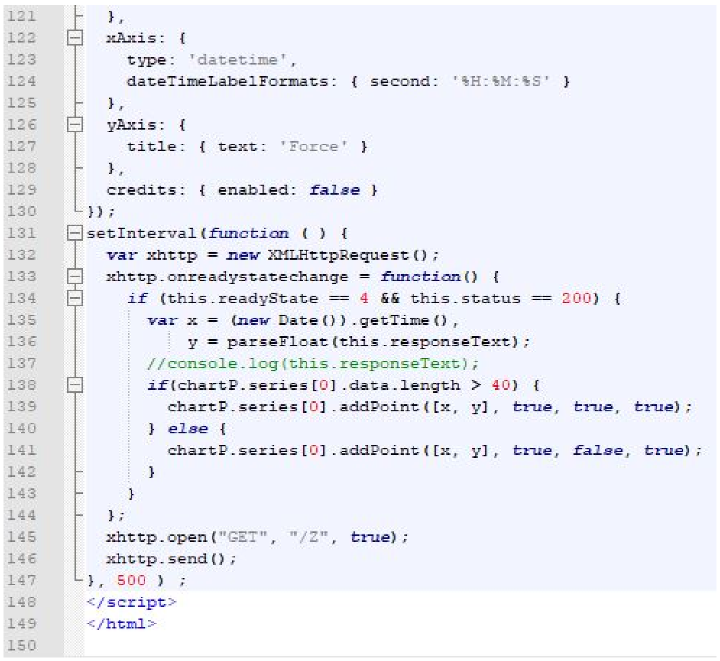
```html
<!DOCTYPE HTML><html>
<!-- Azraful Awal Hamim

 -->
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script src="https://code.highcharts.com/highcharts.js"></script>
  <style>
    body {
      min-width: 310px;
        max-width: 800px;
        height: 400px;
      margin: 0 auto;
    }
    h2 {
      font-family: Arial;
      font-size: 2.5rem;
      text-align: center;
    }
  </style>
</head>
<body>
  <h2>MLX90393 Sensor Readings</h2>
  <div id="chart-X" class="container"></div>
  <div id="chart-Y" class="container"></div>
  <div id="chart-Z" class="container"></div>
</body>
<script>
var chartT = new Highcharts.Chart({
  chart:{ renderTo : 'chart-X' },
  title: { text: 'MLX90393 Sensor X-Axis' },
  series: [{
    showInLegend: false,
    data: []
  }],
  plotOptions: {
    line: { animation: false,
      dataLabels: { enabled: true }
    },
    series: { color: '#059e8a' }
  },
```



```javascript
  },
  xAxis: { type: 'datetime',
    dateTimeLabelFormats: { second: '%H:%M:%S' }
  },
  yAxis: {
    title: { text: 'Force' }

  },
  credits: { enabled: false }
});

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var x = (new Date()).getTime(),
          y = parseFloat(this.responseText);
      //console.log(this.responseText);
      if(chartT.series[0].data.length > 40) {
        chartT.series[0].addPoint([x, y], true, true, true);
      } else {
        chartT.series[0].addPoint([x, y], true, false, true);
      }
    }
  };
  xhttp.open("GET", "/X", true);
  xhttp.send();
}, 500 ) ;

var chartH = new Highcharts.Chart({
  chart:{ renderTo:'chart-Y' },
  title: { text: 'MLX90393 Sensor Y-Axis' },
  series: [{
    showInLegend: false,
    data: []
  }],
  plotOptions: {
    line: { animation: false,
      dataLabels: { enabled: true }
    }
  },
```



```
  xAxis: {
    type: 'datetime',
    dateTimeLabelFormats: { second: '%H:%M:%S' }
  },
  yAxis: {
    title: { text: 'Force' }
  },
  credits: { enabled: false }
});
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var x = (new Date()).getTime(),
          y = parseFloat(this.responseText);
      //console.log(this.responseText);
      if(chartH.series[0].data.length > 40) {
        chartH.series[0].addPoint([x, y], true, true, true);
      } else {
        chartH.series[0].addPoint([x, y], true, false, true);
      }
    }
  };
  xhttp.open("GET", "/Y", true);
  xhttp.send();
}, 500 ) ;

var chartP = new Highcharts.Chart({
  chart:{ renderTo:'chart-Z' },
  title: { text: 'MLX90393 Sensor Z-Axis' },
  series: [{
    showInLegend: false,
    data: []
  }],
  plotOptions: {
    line: { animation: false,
      dataLabels: { enabled: true }
    },
    series: { color: '#18009c' }
```



```
  },
  xAxis: {
    type: 'datetime',
    dateTimeLabelFormats: { second: '%H:%M:%S' }
  },
  yAxis: {
    title: { text: 'Force' }
  },
  credits: { enabled: false }
});
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var x = (new Date()).getTime(),
          y = parseFloat(this.responseText);
      //console.log(this.responseText);
      if(chartP.series[0].data.length > 40) {
        chartP.series[0].addPoint([x, y], true, true, true);
      } else {
        chartP.series[0].addPoint([x, y], true, false, true);
      }
    }
  };
  xhttp.open("GET", "/Z", true);
  xhttp.send();
}, 500 ) ;
</script>
</html>
```