GRADUATION PROJECT 2022

Alexandria University

Faculty Of Engineering

Electrical Engineering Department

Communication And Electronics Section

Smart Healthcare System

Under supervision of

Prof. Dr. El-Sayed Youssef

SUBMITTED BY

Alaa Mohamed Morsy	4
Bassam Mohamed Mahmoud	64
Omar Ahmed Ali Ahmed	126
Fatma Amr Abdul-Mo’men	140
Fatma Mohamed Ahmed	141
Mazen Adel Abbas	149
Noran Essam Abdel-Samea	239
Nourhan Ayman Wageeh	240

Acknowledgement

Firstly, we wish to express our sincere gratitude to Allah who gave us the ability to accomplish this work. This project would not have been possible without the support of many people who have both directly and indirectly contributed to this work .We appreciate with deep respect and sincere gratitude to our supervisor Prof. El Sayed Youssef for his great efforts from starting point till now, His valuable advice which enriched this work and made it more valuable. And finally, we are grateful to our Families for their unparalleled support and continuous encouragement throughout this journey.

Table of Content

List of Figures

List of Tables

List of Abbreviations

ABBREVIATION	
ACK/NACK	Acknowledgment/Negative-Acknowledgment
ADC	Analogue to Digital Converter
API	Application Programming Interface
ARM	Advanced RISC Machine
BPM	Beats per minute
CISC	Complex Instruction Set Computer
CPU	Central Processing Unit
ECG	Electrocardiogram
EEPROM	Electrical Erasable Programmable Read-Only Memory
GND	Ground
GPIO	General Purpose Input Output
gyro	Gyroscope
I/O	Input/Output
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
IoT	Internet of Things
ISA	Instruction Set Architecture
IT	Information Technology

LAN	Local Area Network
LED	Light Emitting Diode
MCU	Microcontroller Unit
MEMS	Micro Electro-Mechanical Systems
MPU	Microprocessor Unit
OS	Operation System
PaaS	Platform as a Service
RAM	Random-Access Memory
RISC	Reduced Instruction Set Computers
ROM	Read-Only Memory
RPi	Raspberry Pi
SCL	Serial Clock
SDA	Serial Data
SDK	Software Development Kit
SMS	Short Message Service
SoC	system-on-chip
SPI	Serial Peripheral Interface
SpO2	Oxygen Saturation
SPS	Samples Per Second
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity

The Internet of Things technologies are the most enabling communication paradigm that could gather, record, analyze the patients' data more accurately and share the knowledge directly among the healthcare service providers. Since Elderly's population percent in 2021 is 5.4 % (expected to be 15% by 2100) and regarding the concept of healthy ageing. It's therefore important to have an All-in-one application for Regular checkups and emergency help old people.

Overview

The Concept of healthy ageing is relatively new to our part of the world even though it has been an established entity in developed world as a part of ageing. [1]

This concept embraces the idea of ageing with least morbidity - the condition of suffering from a disease - and disability - condition that limits a person's movements, senses, or activities -. The spread of certain disease increases with age. The longevity of life depends on the negative significance of the morbidities the population have. [1]

According to research approximately a quarter of recorded deaths can be prevented with proper healthcare. This is because people have no time for themselves and forget about their health management. The reason behind making this project is the growing world of technology and the need for regular health check-ups. Internet of Things (IoT) makes our life easier so, we have decided to make an IoT-based healthcare project for people which provides all their health information and their mobile phones. The best part of this project is that it can be used by everyone and make our health management easier than available systems.

The IoT Health Care System is a healthcare device based on the IoT platform for the patients and doctors. It provides measurements of body parameters like Electrocardiogram (ECG), Temperature, Oxygen saturation, Heartbeat and more.

It also detects the body condition and location of the patients. In this project we are using various sensors and modules for performing different functions. the Cloud is used for storing all the data, it provides security and facility of accessing all the parameters at any time which is very useful for the doctors. This system generates alerts at any critical condition and notifications on fall detection.

The target of our project is helping old people (65 years or above) with chronic diseases who lives in home with a caregiver staying with him to do their regular checkup and early detect any disease related to imbalance.

Introduction

The growing rate of the aging population has brought about many challenges in healthcare service. For example, the service of after stroke rehabilitation for the elderly is an emerging challenge, which requires a long-time commitment of medical and human resources. [2]

In addition, medical errors are still the 3rd leading cause of death after heart disease and cancer All while healthcare costs are continuing to climb.

The core problem here is that elderly patients suffer from chronic diseases that make it so hard for them to do much physical work, even walking from a room to another is difficult, so our device aims to make it easy for this segment of patients to do checkup as if they are in hospital and with such a link with their doctors they can confirm their states according to the output values of our device

One promising method to alleviate the problem is to adopt the IoT technologies and smarten medical service systems. In recent years, applying Internet-based technologies for rehabilitation services has become popular after introducing some new concepts, such as Smarter Planet and Smart City. With the smart perception within an IoT, smart cities can improve the performance of public services and business infrastructure in the ways that real-time data can be collected and analyzed promptly, abrupt, and emergent events can be acknowledged and responded timely, and resources in the cities can be managed and controlled appropriately. As far as the healthcare services, such as medical rehabilitation, are concerned, an IoT-based system makes it possible to provide ‘one stop’ service to the residents conveniently even at remote locations. In contrast to conventional on-site rehabilitation service at local hospitals, all the related resources are shared within communities through smart rehabilitation to provide flexible and convenient treatment to patients. In this way, the utilization of rehabilitation resources can be maximized, and it can be anticipated that the IoT-based intelligent technology would become an irreplaceable tool in modern healthcare systems. [2]

Numerous progresses have been made in healthcare monitoring and control, interoperability and security, pervasive healthcare, and drug interaction checking, etc. These achievements have demonstrated the effectiveness and promising future of IoT-based healthcare system. Despite the existent success, ambiguity and technical challenge still exist regarding the question of how to rapidly and systematically establish as well as deploy an intelligent IoT-based healthcare system that involves big data management. [2]

CHAPTER 1

Internet of Things

IoT

Internet of Things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

How does IoT work?

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors, and communication hardware, to collect, send and act on data they acquire from their environments.

IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices, to set them up, give them instructions or access the data. [3]

The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IoT applications deployed.
IoT can also make use of artificial intelligence (AI) and machine learning to aid in making data collecting processes easier and more dynamic. [3]

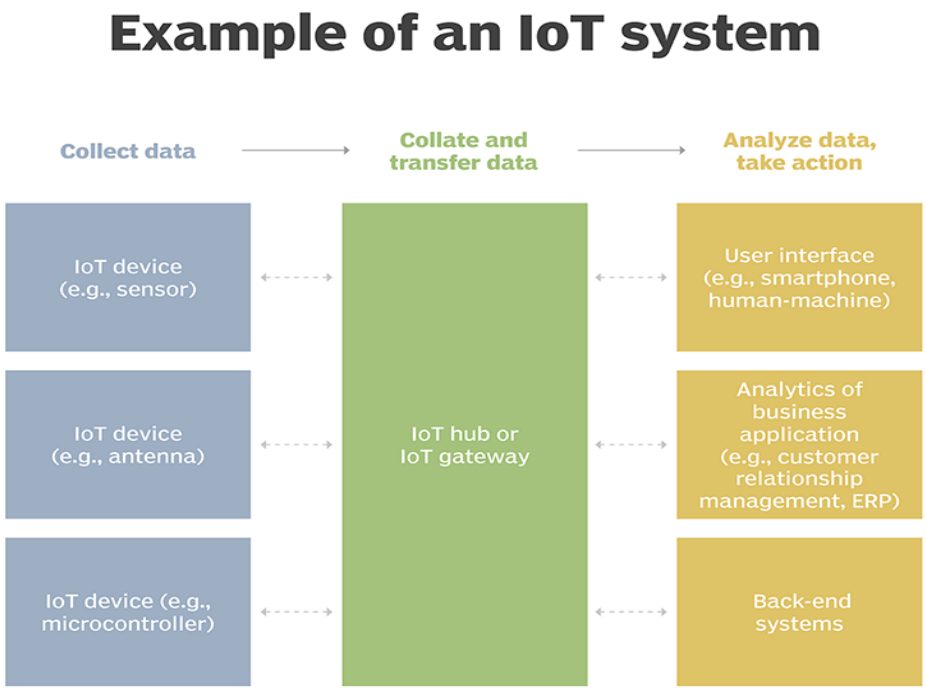


Figure 1 shows an example of an IoT system

Architecture of Internet of Things (IoT)

1. **Sensing Layer –**

   Sensors, actuators, devices are present in this Sensing layer. These Sensors or Actuators accepts data(physical/environmental parameters), processes data and emits data over network.

2. **Network Layer –**

   Internet/Network gateways, Data Acquisition System (DAS) are present in this layer. DAS performs data aggregation and conversion function (Collecting data and aggregating data then converting analog data of sensors to digital data etc). Advanced gateways which mainly opens up connection between Sensor networks and Internet also performs many basic gateway functionalities like malware protection, and filtering also sometimes decision making based on inputted data and data management services, etc.

3. **Data processing Layer –**

   This is processing unit of IoT ecosystem. Here data is analyzed and pre-processed before sending it to data center from where data is accessed by software applications often termed as business applications where data is monitored, and managed and further actions are also prepared. So here Edge IT or edge analytics comes into picture.

4. **Application Layer –**

   This is last layer of 4 stages of IoT architecture. Data centers or cloud is management stage of data where data is managed and is used by end-user applications like agriculture, health care, aerospace, farming, defense, etc.

Why is IoT important?

IoT helps people live and work smarter, as well as gain complete control over their lives. In addition to offering smart devices to automate homes, IoT is essential to business. IoT provides businesses with a real-time look into how their systems really work, delivering insights into everything from the performance of machines to supply chain and logistics operations. [3]

IoT enables companies to automate processes and reduce labor costs. It also cuts down on waste and improves service delivery, making it less expensive to manufacture and deliver goods, as well as offering transparency into customer transactions. [3]

As such, IoT is one of the most important technologies of everyday life, and it will continue to pick up steam as more businesses realize the potential of connected devices to keep them competitive.

IoT for healthcare

Before IoT, patients' interactions with doctors were limited to visits, and tele and text communications. There was no way doctors or hospitals could monitor patients' health continuously and make recommendations accordingly.

IoT-enabled devices have made remote monitoring in the healthcare sector possible, unleashing the potential to keep patients safe and healthy, and empowering physicians to deliver superlative care. It has also increased patient engagement and satisfaction as interactions with doctors have become easier and more efficient. Furthermore, remote monitoring of patient's health helps in reducing the length of hospital stay and prevents re-admissions. IoT also has a major impact on reducing healthcare costs significantly and improving treatment outcomes. [4]

IoT is undoubtedly transforming the healthcare industry by redefining the space of devices and people interaction in delivering healthcare solutions. IoT has applications in healthcare that benefit patients, families, physicians, hospitals, and insurance companies.

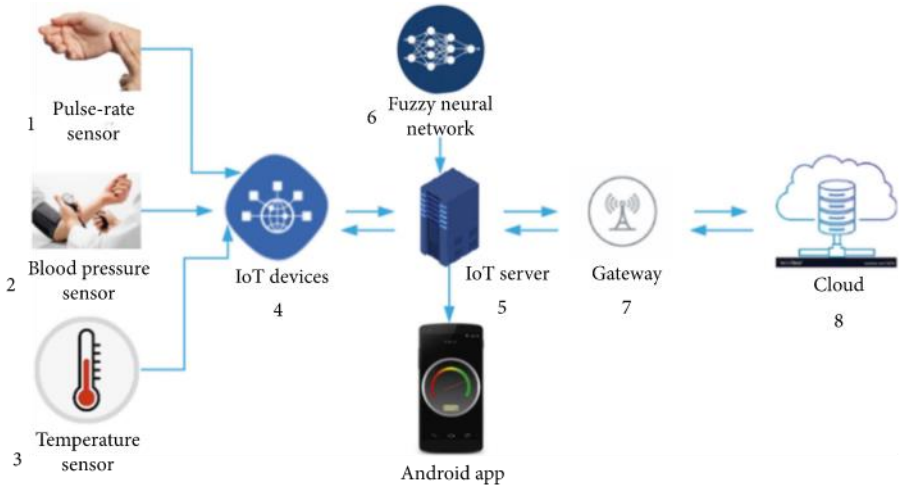


Figure 2 shows an example of an IoT system for healthcare

IoT for Patients

Devices in the form of wearables like fitness bands and other wirelessly connected devices like blood pressure and heart rate monitoring cuffs, glucometer etc. give patients access to personalized attention. These devices can be tuned to remind calorie count, exercise check, appointments, blood pressure variations and much more.

IoT has changed people's lives, especially elderly patients, by enabling constant tracking of health conditions. This has a major impact on people living alone and their families. On any disturbance or changes in the routine activities of a person, alert mechanism sends signals to family members and concerned health providers.

IoT for Physicians

By using wearables and other home monitoring equipment embedded with IoT, physicians can keep track of patients' health more effectively. They can track patients' adherence to treatment plans or any need for immediate medical attention. IoT enables healthcare professionals to be more watchful and connect with the patients proactively. Data collected from IoT devices can help physicians identify the best treatment process for patients and reach the expected outcomes. [5]

IoT for Hospitals

Apart from monitoring patients' health, there are many other areas where IoT devices are very useful in hospitals. IoT devices tagged with sensors are used for tracking real time location of medical equipment like wheelchairs, defibrillators, nebulizers, oxygen pumps and other monitoring equipment. Deployment of medical staff at different locations can also be analyzed real time.

The spread of infections is a major concern for patients in hospitals. IoT-enabled hygiene monitoring devices help in preventing patients from getting infected. IoT devices also help in asset management like pharmacy inventory control, and environmental monitoring, for instance, checking refrigerator temperature, and humidity and temperature control. [5]

IoT for Health Insurance Companies

There are numerous opportunities for health insurers with IoT-connected intelligent devices. Insurance companies can leverage data captured through health monitoring devices for their underwriting and claims operations. This data will enable them to detect fraud claims and identify prospects for underwriting. [5]

IoT devices bring transparency between insurers and customers in the underwriting, pricing, claims handling, and risk assessment processes. [5] In the light of IoT-captured data-driven decisions in all operation processes, customers will have adequate visibility into underlying thought behind every decision made and process outcomes. [6]

Redefining Healthcare

The proliferation of healthcare specific IoT products opens immense opportunities. And the huge amount of data generated by these connected devices hold the potential to transform healthcare.

IoT has a four-step architecture that are basically stages in a process (See Figure 3). All four stages are connected in a manner that data is captured or processed at one stage and yields the value to the next stage. Integrated values in the process brings intuitions and deliver dynamic business prospects.

Step 1:
First step consists of deployment of interconnected devices that includes sensors, actuators, monitors, detectors, camera systems etc. These devices collect the data.

Step 2:
Usually, data received from sensors and other devices are in analog form, which need to be aggregated and converted to the digital form for further data processing.

Step 3:
Once the data is digitized and aggregated, this is pre-processed, standardized, and moved to the data center or Cloud.

Step 4:
Final data is managed and analyzed at the required level. Advanced Analytics, applied to this data, brings actionable business insights for effective decision-making.

IoT is redefining healthcare by ensuring better care, improved treatment outcomes and reduced costs for patients, and better processes and workflows, improved performance, and patient experience for healthcare providers.

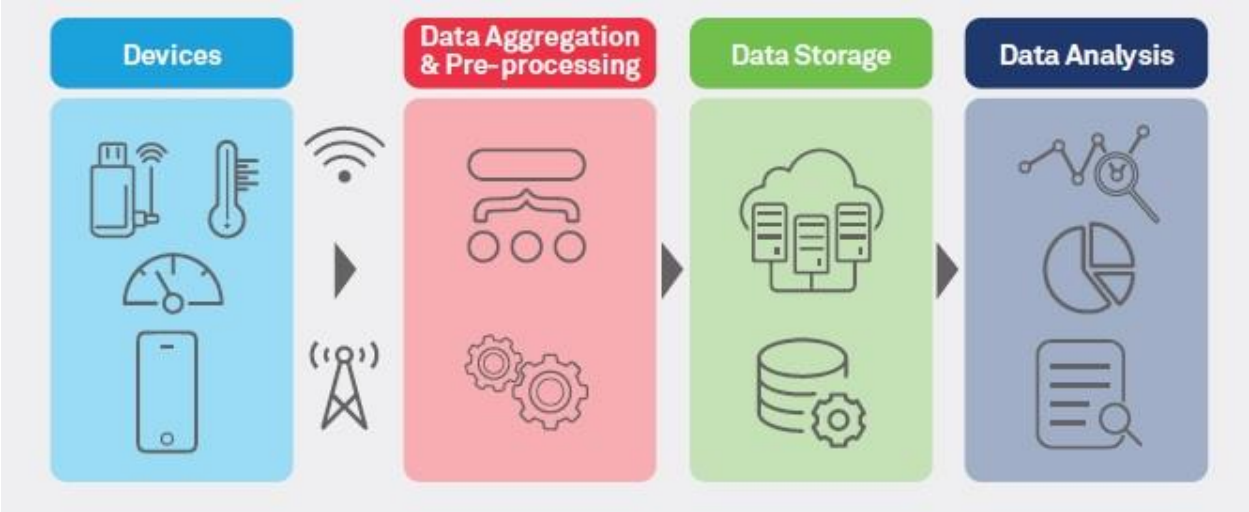


Figure 3 shows an IoT system Architecture

The major advantages of IoT in healthcare

- **Cost Reduction**: IoT enables patient monitoring in real time, thus significantly cutting down unnecessary visits to doctors, hospital stays and re-admissions.
- **Improved Treatment:** It enables physicians to make evidence-based informed decisions and brings absolute transparency.
- **Faster Disease Diagnosis:** Continuous patient monitoring and real time data helps in diagnosing diseases at an early stage or even before the disease develops based on symptoms. [5]
- **Proactive Treatment**: Continuous health monitoring opens the doors for providing proactive medical treatment.
- **Drugs and Equipment Management:** Management of drugs and medical equipment is a major challenge in a healthcare industry. Through connected devices, these are managed and utilized efficiently with reduced costs. [5]
- **Error Reduction:** Data generated through IoT devices not only help in effective decision making but also ensure smooth healthcare operations with reduced errors, waste, and system costs. [5]

Healthcare IoT is not without challenges. IoT-enabled connected devices capture huge amounts of data, including sensitive information, giving rise to concerns about data security.

Implementing apt security measures is crucial. IoT explores new dimensions of patient care through real-time health monitoring and access to patients' health data. This data is a goldmine for healthcare stakeholders to improve patient's health and experiences while making revenue opportunities and improving healthcare operations. Being prepared to harness this digital power would prove to be the differentiator in the increasingly connected world.

CHAPTER 2

Embedded System

A Microcontroller is the word that comes to mind on talking about an Embedded System device but is it the suitable for our project, in this chapter we will discuss if the controller we choose to orchestrate our project.

What is a Microcontroller?

A microcontroller (sometimes called an MCU or Microcontroller Unit) is a single Integrated Circuit (IC) that is typically used for a specific application and designed to implement certain tasks. Products and devices that must be automatically controlled in certain situations, like appliances, power tools, automobile engine control systems, and computers are great examples, but microcontrollers reach much further than just these applications.

Essentially, an MCU gathers input, processes this information, and outputs a certain action based on the information gathered. Microcontrollers usually operate at lower speeds, around the 1MHz to 200 MHz range, and need to be designed to consume less power because they are embedded inside other devices that can have greater power consumptions in other areas.

A microcontroller can be seen as a small computer, and this is because of the essential components inside of it; the Central Processing Unit (CPU), the Random-Access Memory (RAM), the Flash Memory, the Serial Bus Interface, the Input/Output Ports (I/O Ports), and in many cases, the Electrical Erasable Programmable Read-Only Memory (EEPROM).

The CPU, sometimes called a processor or microprocessor, controls all of the instructions/data flow that it receives. You can think of it as the brains of the system, processing all the data input it receives and executes the required instructions. [7]

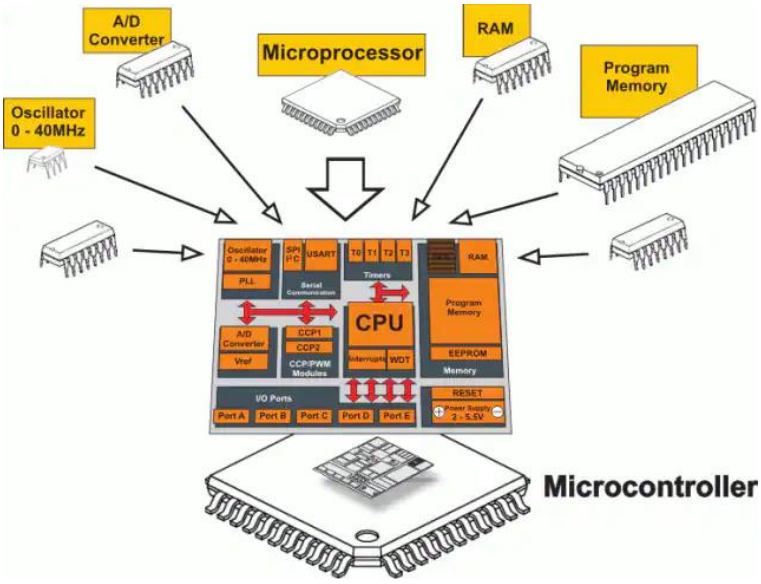


Figure 4 shows a diagram of the main parts and other parts in the microcontroller.

Now, the following table highlights the differences between a microprocessor and a microcontroller [8]:

Table 1 Differences between a Microprocessor and a Microcontroller

	Microprocessor	**Microcontroller**
Application	It used where intensive processing is required. It is used in personal computers, laptops, mobiles, video games, etc.	It used where the task is fixed and predefined. It is used in the washing machine, alarm, etc.
Structure	It has only the CPU in the chip. Other devices like I/O port, memory, timer is connected externally. The structure of the microprocessor is flexible. Users can decide the amount of memory, the number of I/O port and other peripheral devices.	CPU, Memory, I/O port, and all other devices are connected on the single chip. The structure is fixed. Once it is designed the user cannot change the peripheral devices.
Clock speed	The clock speed of the microprocessor is high. It is in terms of the GHz. It ranges between 1 GHz to 4 GHz.	The clock speed of the microcontroller is less. It is in terms of the MHz it ranges between 1 MHz to 300 MHz.
RAM	The volatile memory (RAM) for the microprocessor is in the range of the 512 MB to 32 GB.	The volatile memory (RAM) for the microcontroller is in the range of 2 KB to 256 KB.
ROM	The hard disk (ROM) for the microprocessor is in the range of the 128 GB to 2 TB.	The hard drive or flash memory (ROM) is in the range of the 32 KB to 2 MB.
Peripheral interface	The common peripheral interface for the microprocessor is USB, UART, and high-speed Ethernet.	The common peripheral interface for the microcontroller is I2C, SPI, and UART.
Bit size	It is available in 32-Bit and 64-bit.	It is available in 8-bit, 16-bit, and 36-bit.
Cost	The cost of the microprocessor is high compared to the microcontroller.	It is cheaper.
Power consumption	The power consumption for the microprocessor is high.	The power consumption for the microcontroller is less.
Size	The overall size of the system is large.	The overall size of the system is small.

In our project, we need two controllers. The first must have high processing ability, high speed, and ability to use Ethernet and Wi-Fi. The Second also need high speed and the Wi-Fi using ability but it does not need that high processing capability as the first.

So regarding *Table 1*, we need a Microprocessor but this will make the project's cost very expensive and become intensive power consumer!!

That makes us search for a device that provides the microprocessor properties but in cheaper cost and not power consuming which is **System-on-Chip**.

What is System-on-Chip?

A System on Chip (usually known as an SoC) is basically a circuit embedded on a small coin-sized chip and integrated with a microcontroller or microprocessor. A SoC is an encapsulation of one or more of CPUs, microcontrollers, other accelerators or supporting hardware and more specifically it does not have a specific standard about what type of circuitry it should contain. [9]

The components that an SoC generally looks to incorporate within itself include a central processing unit, I/O ports, internal memory, as well as analog input and output blocks among other things. Depending on the kind of system that has been reduced to the size of a chip, it can perform a variety of functions including signal processing, wireless communication, artificial intelligence and more. [10]

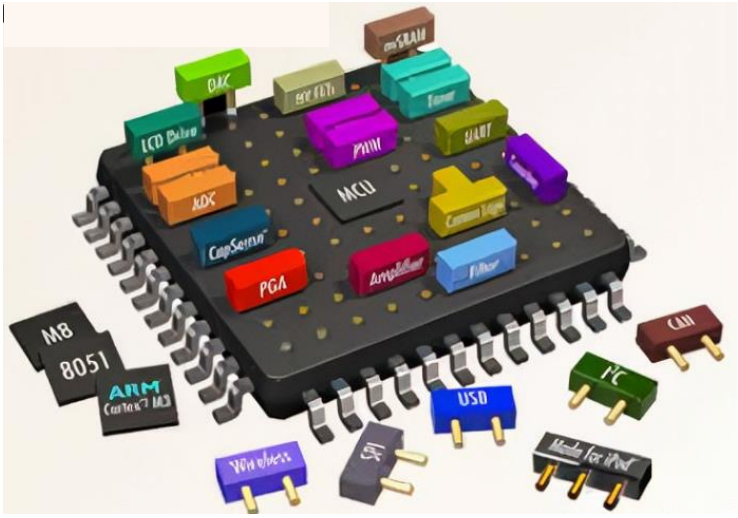


Figure 5 shows the System-on-Chip components

Why use a System-on-Chip?

SoC devices are reliable, customizable, and relatively cost-effective. The heart of systems on a chip technology is a microprocessor core or microcontroller core. One of the main motivators behind the creation of systems on a chip is the fact that moving ahead into the future, its primary goal is to reduce energy waste, save up on spending costs, as well as reduce the space occupied by large systems.

With an SoC, you achieve all those goals as you essentially size down what is normally multichip designs onto a single processor that uses much less power than before. We can carry it anywhere and everywhere with us without ever having to compromise on the capability and functionality of the gadgets. As such, they are frequently used in systems pertaining to the Internet of Things, embedded systems, as well as our own smartphones, cars and more. [10]

The application of SoCs in the practical world are practically limitless and priceless. They are used in most, if not all, portable tech such as smartphones, cameras, tablets, and other wireless technologies. Your smartphone is a good example of how a system on chip works. When you use your cell phone, you do not only use it to make and receive calls- you also use it to browse the internet, view videos, listen to audio, take photos, play games, text message, and whatnot. None of this would be possible without having multiple components such as a graphics card, internet support, wireless connections, GPS, and many other elements. An SoC allows you to take all these components, put them on a single chip, shrink it down to a size that can fit in the palm of your hand, and carry it around as a living and breathing system in your phone. [10]

We chose using Raspberry Pi 3 model B - programmed in python- and NodeMCU-programmed with Arduino Integrated Development Environment (IDE) -as they cover the project needed features. We will discuss them in this chapter.

Raspberry Pi

What is Raspberry Pi?

Raspberry Pi (known as RPi, Raspi), developed by Raspberry Pi Foundation in association with Broadcom, is a series of small single-board computers and perhaps the most inspiring computer available today.

The first model of the RPi was released in 2012, and as of 2021 there have been five generations of the boards (See Appendix A). In addition, an MCU called the Pico was released in early 2021.

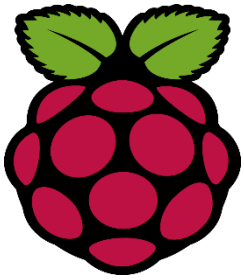


Figure 6 shows Raspberry Pi Foundation logo

All RPi models have one thing in common, though: they are compatible, meaning that software written for one model will run on any other model.

The processor at the heart of the RPi system is a Broadcom BCM2835/ BCM2837 SoC multimedia processor. This means that the vast majority of the system's components, including its CPU and graphics processing unit (GPU) along with the audio and communications hardware, are built onto that single component hidden beneath the memory chip at the center of the board.

This SoC uses an ARM instruction set architecture (ISA). Its combination of a RISC architecture and low-power draw make it the perfect choice over desktop chips with high power demands and CISC architectures. The BCM2835 uses a generation of ARM's processor design known as ARM11, which in turn is designed around a version of the ISA known as ARMv6. Then it has arrived in the more advanced ARMv7 architecture which the ARM Cortex family of processors and the RPi generations currently use. [11]

RPi supports many programming languages as C/C++, Python 2/3, and Scratch by default. However, nearly any language compiler or interpreter can be installed on Raspberry Pi OS (previously known as Raspbian).

Raspberry Pi 3 Model B

Raspberry Pi 3 Model B is a Single-board computer with wireless Local Area Network (LAN) and Bluetooth connectivity, the earliest model of the third-generation Raspberry Pi.

Figure 7 shows Raspberry Pi 3 Model B Development board

Specifications

- Quad Core 1.2GHz Broadcom BCM2837 64bit SoC
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended General Purpose Input Output (GPIO)
- 4 Universal Serial Bus (USB) 2 ports
- 4 Pole stereo output and composite video port
- Full size High-Definition Multimedia Interface (HDMI)
- Camera Serial Interface (CSI) camera port for connecting a RPi camera
- Display Serial Interface (DSI) display port for connecting a RPi touchscreen display
- Micro SD (Secure Digital) port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A [12]

Raspberry Pi-3 Pinout

Figure 8 shows Raspberry Pi Pinout

Raspberry Pi-3 Pinout Configuration

Table 2 Raspberry Pi-3 Pinout Configuration

PIN GROUP	PIN NAME	DESCRIPTION [13]
Power source	+5V, +3.3V, GND and Vin	+5V -power output +3.3V -power output GND – GROUND pin
Communication interface	UART Interface (RXD, TXD)	UART used for interfacing sensors and other devices.
	SPI Interface (MOSI, MISO, CLK, CE) x 2	SPI used for communicating with other boards or peripherals.
	I2C Interface (SDA, SCL) x 2	I2C Interface can be used to connect peripherals.
Input /output pins	26 I/O	some pins have multiple functions they can be considered as I/O pins.
Pulse Width Modulation (PWM)	Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19	These 4 channels can provide PWM outputs.
External interrupts	All I/O	In the board all I/O pins can be used as Interrupts.

NodeMCU

Figure 9 shows NodeMCU Platform logo

What is NodeMCU?

The NodeMCU (Node MicroController Unit) is an open source software based on Lua firmware and a hardware development environment that is built around a very inexpensive SoC called the ESP8266.

The ESP8266, designed and manufactured by Espressif Systems, contains all crucial elements of the modern computer: CPU, RAM, networking Wireless Fidelity (Wi-Fi), and even a modern operating system and Software Development Kit (SDK). [14]

However, as a chip, the ESP8266 is also hard to access and use. we must solder wires, with the appropriate analog voltage, to its PINs for the simplest tasks such as powering it on or sending a keystroke to the "computer" on the chip. [14]

And, we have to program it in low-level machine instructions that can be interpreted by the chip hardware. While this level of integration is not a problem when the ESP8266 is used as an embedded controller chip. [14]

Borrowing a page from the successful playbooks of Arduino or a RPi, the NodeMCU project aims to simplify ESP8266 development. It has two key components.

1. An open source ESP8266 firmware that is built on top of the chip manufacturer's proprietary SDK. The firmware provides a simple programming environment based on eLua (embedded Lua), which is a very simple and fast scripting language with an established developer community.
2. A development kit (DEVKIT) board that incorporates the ESP8266 chip on a standard circuit board. The board has a built-in USB port that is already wired up with the chip, a hardware reset button, Wi-Fi antenna, LED lights, and standard-sized GPIO pins that can plug into a bread board.

Specifications & Features

- Tensilica 32-bit RISC CPU Xtensa LX106 Microprocessor
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- 16 Digital I/O Pins (DIO)
- Analog Input Pins Analogue to Digital Converter
- 2 UARTs,1 SPIs,1 I2Cs
- Flash Memory: 4 MB
- RAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL included
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects
- Easy to use
- Programmability with Arduino IDE
- Available as an access point or station
- practicable in Event-driven Application Programming Interface (API) applications [15, 16]

Figure 10 shows NodeMCU ESP8266 Development board

NodeMCU ESP8266 Pinout

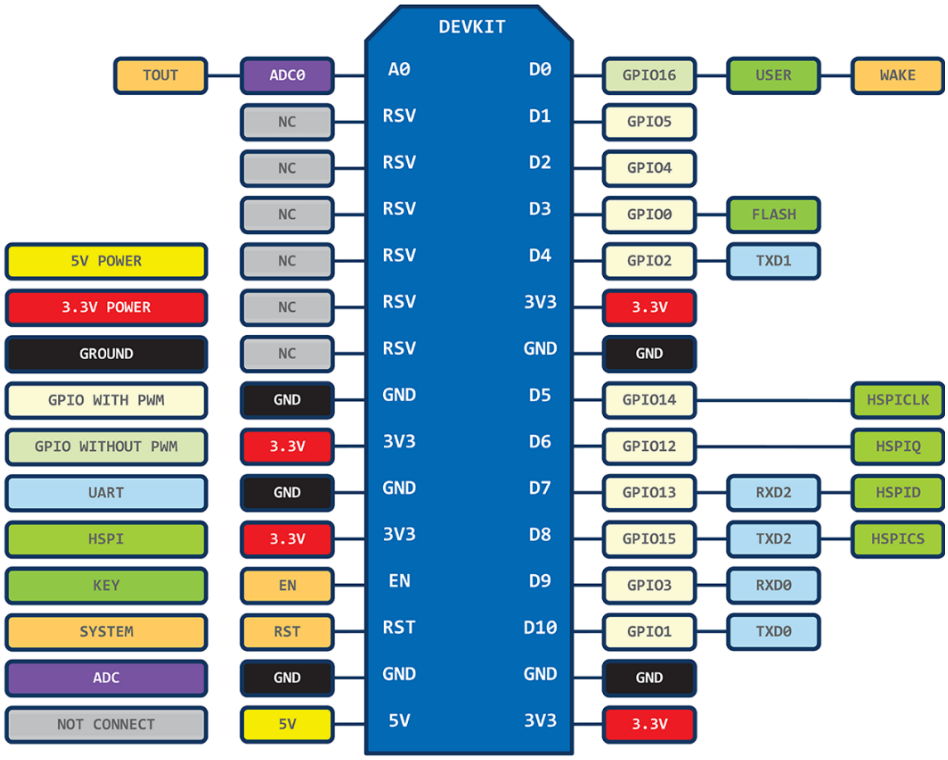


Figure 11 shows NodeMCU Pinout

NodeMCU Development Board Pinout Configuration

Table 3 NodeMCU Development Board Pinout Configuration

Pin Category	Name	Description [16]
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	EN, RST	The pin and the button reset the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

Community Support Raspberry Pi and NodeMCU Comparison

The existence of devices such as the Raspberry Pi and NodeMCU has led to the existence of a significant hobbyist community. Since the kind of sensors used are pretty much standardized, the microcontroller (or in case of the raspberry pi the processor) make all the difference when it comes to factors such as ease of use, power draw, hardware and software support. [17]

The Raspberry Pi is one of the most popular single-board computers (SBCs) out there, however it has less extensive hardware documentation than NodeMCU.

More experienced developers can leverage the power it provides, however for a beginner dealing with Python can be daunting although the Raspberry Pi foundation has been working on gradually bridging the gap.

The board is also supported by online IoT platforms such as My devices Cayenne and Blynk which makes it trivial to get the project onto the cloud. The RPi can also do a significant amount of processing locally and for the price, there is nothing else quite like it. [17]

The NodeMCU brings a unique value proposition with being fairly powerful and cheap. It can run off the Arduino IDE with a few minor modifications and can also be programmed in LUA programing language for those who are more technically inclined.

The built in Wi-Fi module makes it very easy to connect to MQTT servers and to cloud. Community support is rapidly growing and there are several different modules built on both ESP8266 and its more powerful successor, the ESP32. Much like the RPi, it is also compatible with all IoT platforms. [17]

Inter-Integrated Circuit(I2C)

I2C is a serial communication protocol, so data is transferred bit by bit along a single wire -the Serial Date (SDA) line -.
Like SPI, I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal - Serial Clock (SCL) - shared between the master and the slave. The clock signal is always controlled by the master. [18]

How I2C works?

With I2C, data is transferred in *messages.* Messages are broken up into *frames* of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted.
The message also includes start and stop conditions, read/write bits, and ACK/NACK (Acknowledgment/Negative-Acknowledgment) bits between each data frame [18]:

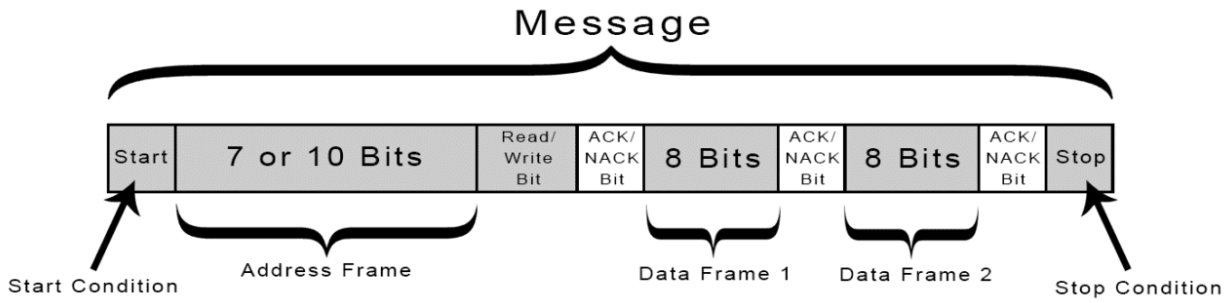


Figure 12 shows I2C Data Frame

Start Condition: The SDA line switches from a high voltage level to a low voltage level *before* the SCL line switches from high to low.

Stop Condition: The SDA line switches from a low voltage level to a high voltage level *after* the SCL line switches from low to high.

Address Frame: A 7- or 10-bit sequence unique to each slave that identifies the slave when the master wants to talk to it.

Read/Write Bit: A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).

ACK/NACK Bit: Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

ADDRESSING

I2C doesn't have slave select lines like SPI, so it needs another way to let the slave know that data is being sent to it, and not another slave. It does this by *addressing*. The address frame is always the first frame after the start bit in a new message.

The master sends the address of the slave it wants to communicate with to every slave connected to it. Each slave then compares the address sent from the master to its own address. If the address matches, it sends a low voltage ACK bit back to the master. If the address doesn't match, the slave does nothing, and the SDA line remains high.

READ/WRITE BIT

The address frame includes a single bit at the end that informs the slave whether the master wants to write data to it or receive data from it. If the master wants to send data to the slave, the read/write bit is a low voltage level. If the master is requesting data from the slave, the bit is a high voltage level.

THE DATA FRAME

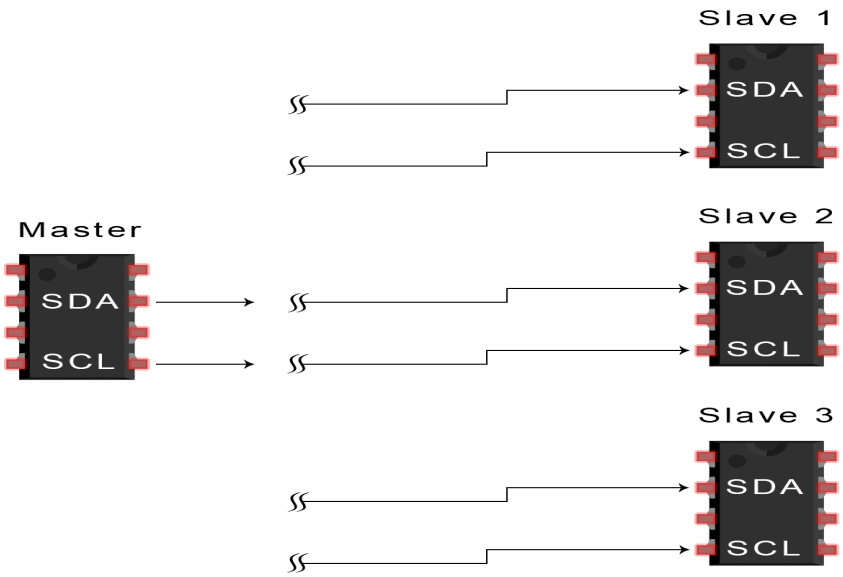
After the master detects the ACK bit from the slave, the first data frame is ready to be sent.

The data frame is always 8 bits long and sent with the most significant bit first. Each data frame is immediately followed by an ACK/NACK bit to verify that the frame has been received successfully. The ACK bit must be received by either the master or the slave (depending on who is sending the data) before the next data frame can be sent.

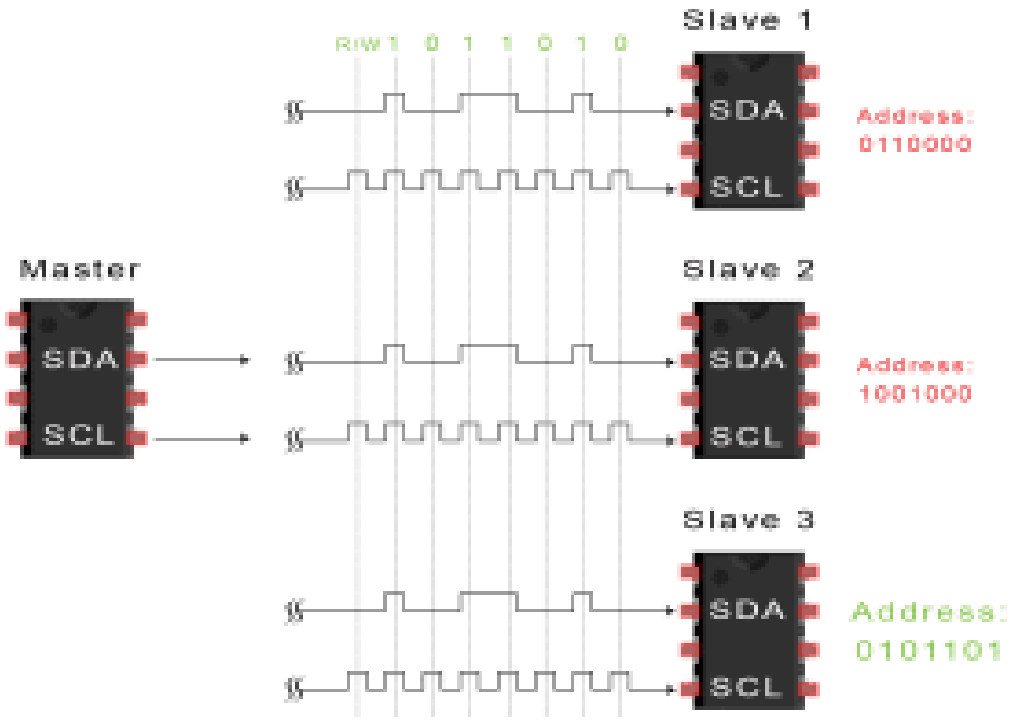
After all of the data frames have been sent, the master can send a stop condition to the slave to halt the transmission. The stop condition is a voltage transition from low to high on the SDA line after a low to high transition on the SCL line, with the SCL line remaining high.

Steps of I2C data transmission

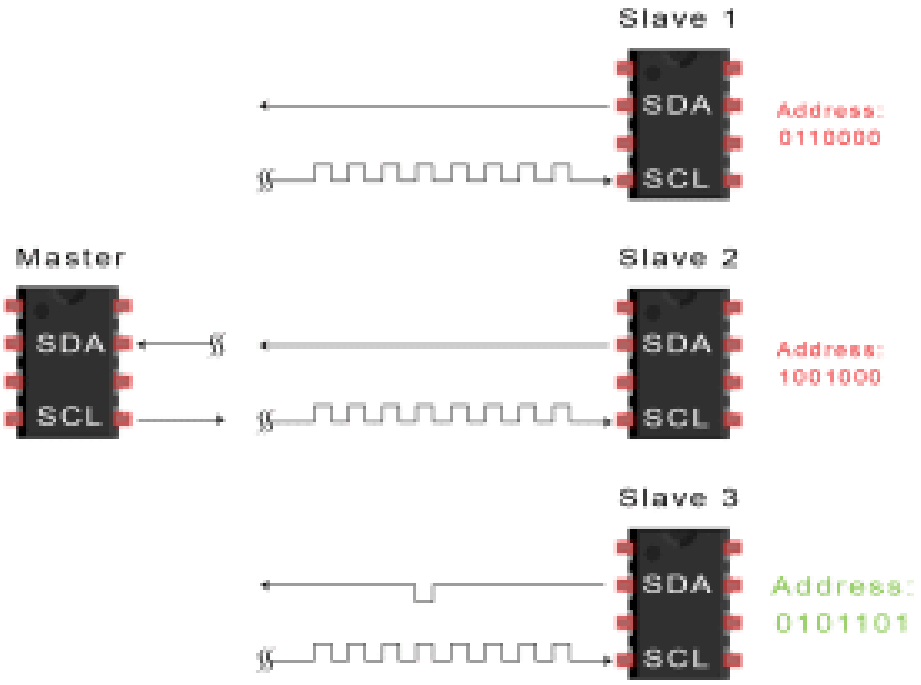
1. The master sends the start condition to every connected slave by switching the SDA line from a high voltage level to a low voltage level before switching the SCL line from high to low. [18]



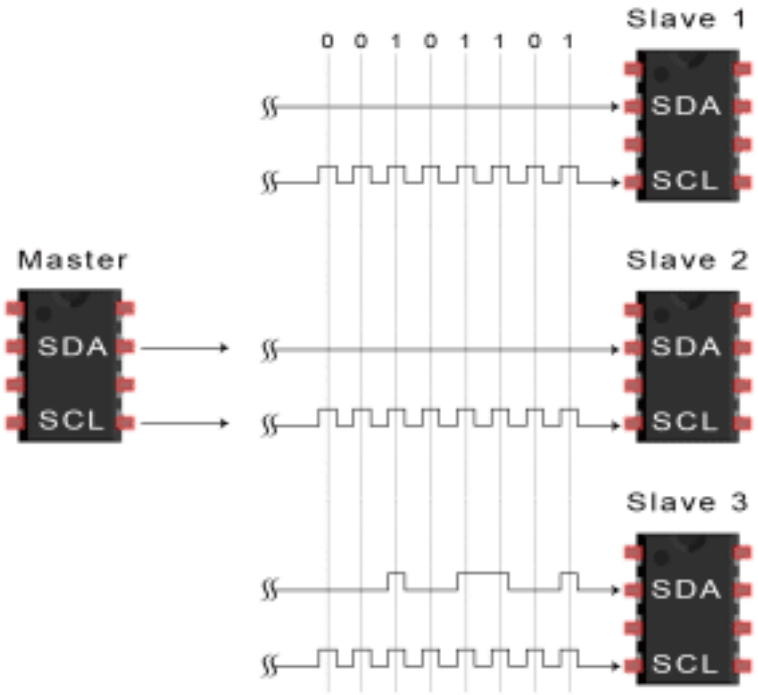
2. The master sends each slave the 7- or 10-bit address of the slave it wants to communicate with, along with the read/write bit:



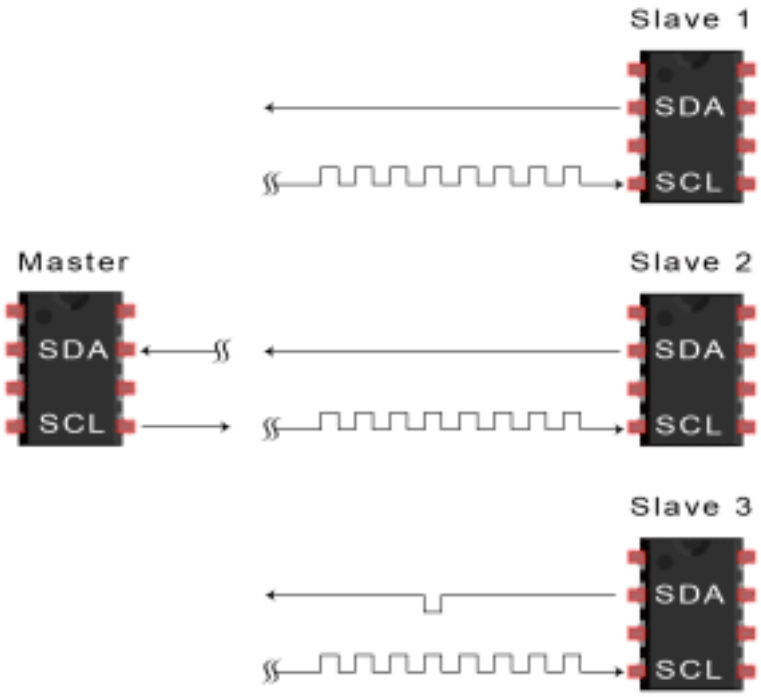
3. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an ACK bit by pulling the SDA line low for one bit. If the address from the master does not match the slave's own address, the slave leaves the SDA line high.



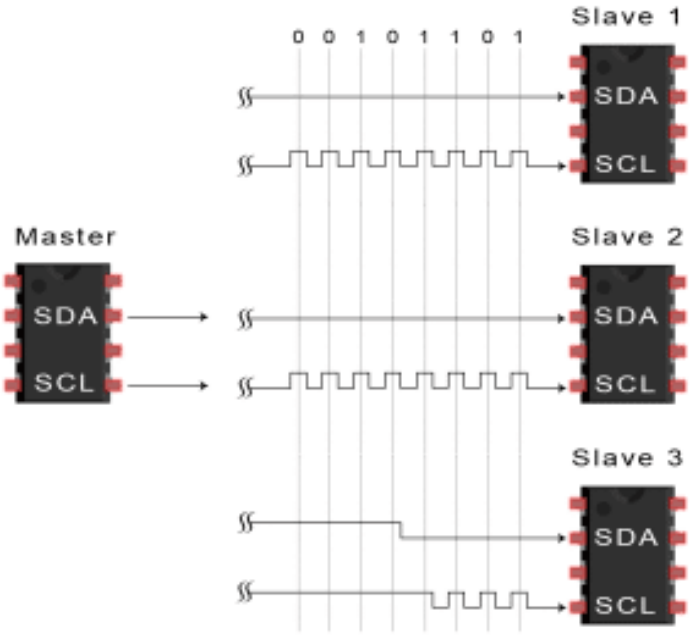
4. The master sends or receives the data frame:



5. After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame:



6. To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching SDA high:



Advantages and Disadvantages of I2C

There is a lot to I2C that might make it sound complicated compared to other protocols, but there are some good reasons why you may or may not want to use I2C to connect to a particular device. [18]

Advantages

- Only uses two wires
- Supports multiple masters and multiple slaves
- ACK/NACK bit gives confirmation that each frame is transferred successfully
- Hardware is less complicated than with UARTs
- Well known and widely used protocol

Disadvantages

- Slower data transfer rate than SPI
- The size of the data frame is limited to 8 bits
- More complicated hardware needed to implement than SPI

CHAPTER 3

Cloud Networking

What is cloud networking?

Cloud networking is a type of Information Technology (IT) infrastructure in which some or all of an organization's network capabilities and resources are hosted in a public or private cloud platform, managed in-house or by a service provider, and available on demand. [19]

Companies can either use on-premises cloud networking resources to build a private cloud network or use cloud-based networking resources in the public cloud, or a hybrid cloud combination of both. These network resources can include virtual routers, firewalls, and bandwidth and network management software, with other tools and functions available as required. [19]

Why cloud networking?

Businesses today turn to the cloud to drive agility, deliver differentiation, accelerate time-to-market, and increase scale. The cloud model has become the standard approach to build and deliver applications for the modern enterprise. [19]

Cloud networking has also played a critical role in the way organizations address their growing infrastructure needs, regional expansions, and redundancy plans. Many organizations are adopting a multi-data center strategy and leveraging multiple clouds from multiple cloud service providers (CSPs).

Benefits of cloud networking

Most organizations have become a patchwork of on-premises technologies, public cloud services, legacy applications and systems, and emerging technologies — a complex situation that contributes to a weak security posture and results in inadequate governance, visibility, and manageability across fragmented networks. [19]

A Virtual Cloud Network is VMware's vision of the future of networking. It is an architectural approach (not a product) built in software at global scale from edge-to-edge, that's able to deliver consistent, pervasive connectivity and security for apps and data wherever they reside, independent of underlying physical infrastructure. [19]

Whether your workloads are on premises or in the cloud, the same network and security stack can be used to provide connectivity, security, and visibility. It is also the kind of next-generation networking service consumption technology that IT is increasingly adopting to provide the digital fabric that helps unify a hyper-distributed world. [19]

Examples of Cloud Computing

Cloud computing is the use of hardware or software off-site that is accessed over networks for computing needs. Examples of cloud computing depend on the type of cloud computing services being provided. [20]

The main types of cloud computing include software as a service, platform as a service, and infrastructure as a service. Serverless computing, also known as function as a service (FaaS), is also a popular method of cloud computing for businesses. [20]

- **SaaS or Software as a Service**.

  SaaS means instead of installing software on your computer, you access the platform online.

  Examples would include:
  - Square, which processes payments online.
  - Google Apps such as Google Drive or Calendar.
  - Slack, which allows collaboration and chat between other users.

- **IaaS or Infrastructure as a Service.**

  IaaS provides infrastructure components such as servers, storage, networking, security, and moreover the cloud.
  Examples would include:
  - Dropbox, a file storage and sharing system.
  - Microsoft Azure, which offers backup and disaster recovery services, hosting, and more.
  - Rackspace, which offers data, security, and infrastructure services.

- **PaaS or Platform as a Service.**

  PaaS provides computing platforms such as operating systems, programming language execution environments, databases, and web servers.

  Examples would include:

  - Google App Engine and Heroku, which allow developers to develop and serve apps.

- **Serverless Computing.**

  Serverless computing (also called simply "Serverless") is simply using a server on the cloud. This offers more elasticity, easier maintenance, and is often more price effective than hosting servers on-site.

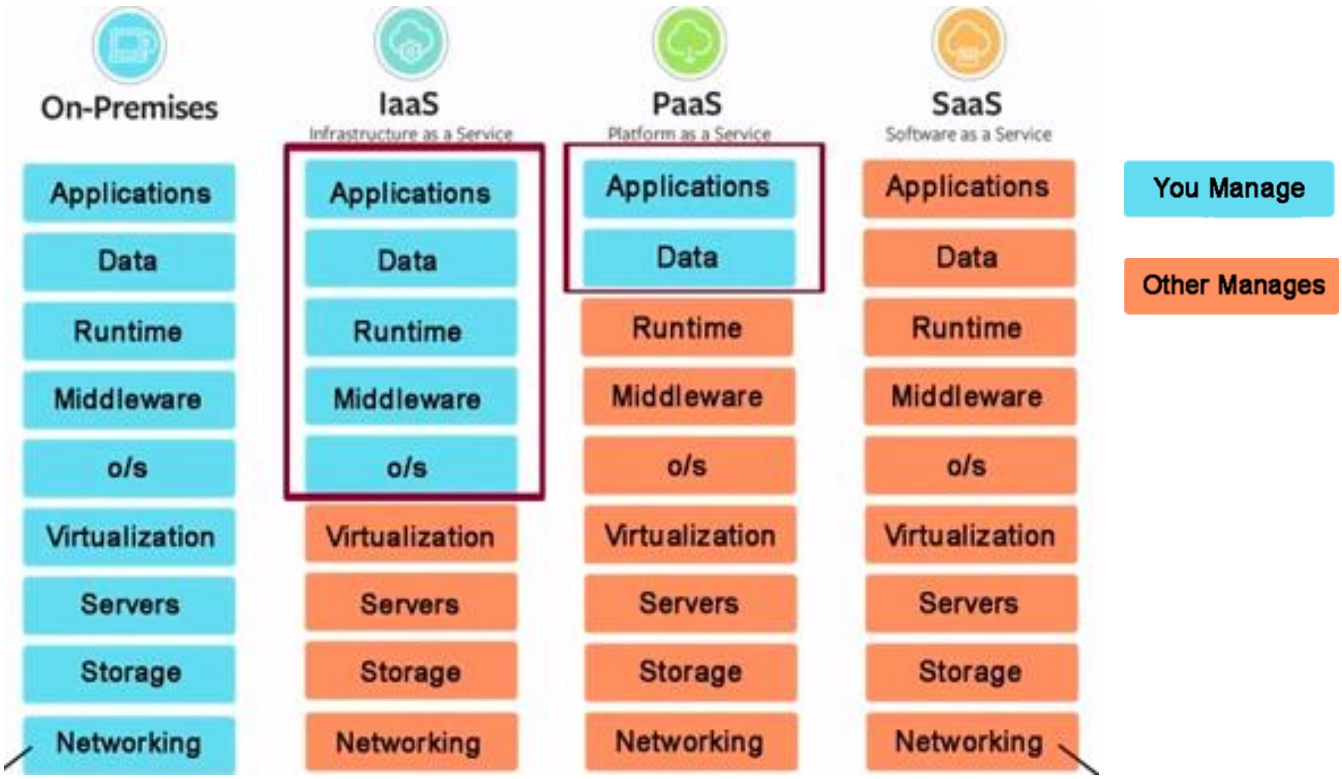
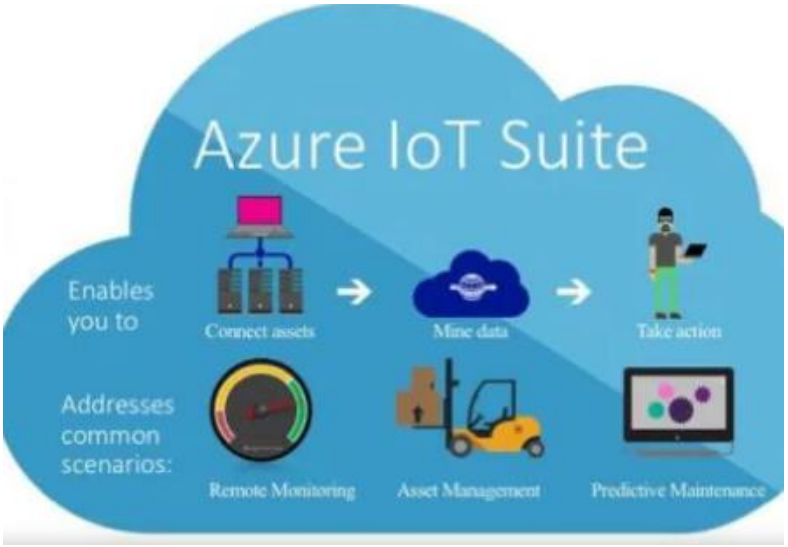


Figure 13 shows Types of Cloud Computing

The most popular clouds used in IoT

1. Microsoft Azure IoT Suite

Microsoft Azure provides multiple services to create IoT solutions. It enhances your profitability and productivity with pre-built connected solutions. It analyzes untapped data to transform business. This provides the solutions for a small Proof-of-Concept (PoC) to Rolling out your ideas. Azure Suite can easily analyze and act on new data. [21]



Azure IoT Suite provides features like:

- Easy Device Registry.
- Rich Integration with Salesforce, Oracle, WebSphere, etc.
- Dashboards and visualization.
- Real-time streaming.

2. **IBM Watson IoT Platform**

IBM Watson is a powerful platform backed by IBM 's the Bluemix and hybrid cloud PaaS development platform. By providing easy sample apps and interfaces for IoT services, they make it accessible to beginners. You can easily try out their sample to see how it works, which makes it stand out from other platforms. [21]

Users can get the following features:

- Real-time data exchange
- Secure Communication
- Cognitive systems
- Recently added data sensor and weather data service

3. **AWS IoT Platform**

Amazon made it much easier for developers to collect data from sensors and Internet-connected devices. They help you collect and send data to the cloud and analyze that information to provide the ability to manage devices. [21]

You can easily interact with your application with the devices even they are offline.

Main features of the AWS IoT platform are:

- Device management
- Secure gateway for devices
- Authentication and encryption
- Device shadow

4. **Blynk IoT Platform**

Blynk is an IoT platform for iOS or Android smartphones that is used to control Arduino, RPi and NodeMCU via the Internet. This application is used to create a graphical interface or human machine interface (HMI) by compiling and providing the appropriate address on the available widgets. [21]

It provides many things such as:

- It can control hardware remotely.
- It can display sensor data.
- It can store data, visualize it, and do many other cool things.

This cloud was supposed to be the chosen one for our project, but we faced some difficulties which was that we couldn't upload its libraries on our microcontroller 'RPi'.

5. **ThingSpeak IoT Platform**

ThingSpeak is an open-source platform that allows you to collect and store sensor data to the cloud. It provides you the app to analyze and visualize your data in MATLAB. You can use Arduino, RPi, and Beaglebone to send sensor data. You can create a separate channel to store data. [21]

Features of ThingSpeak:

- Collect data in private channels
- App integration
- Event scheduling
- MATLAB analytics and visualization

This cloud is the one we chose on dealing with RPi after failing to use Blynk cloud as discussed above, so let's talk more about it and how it works.

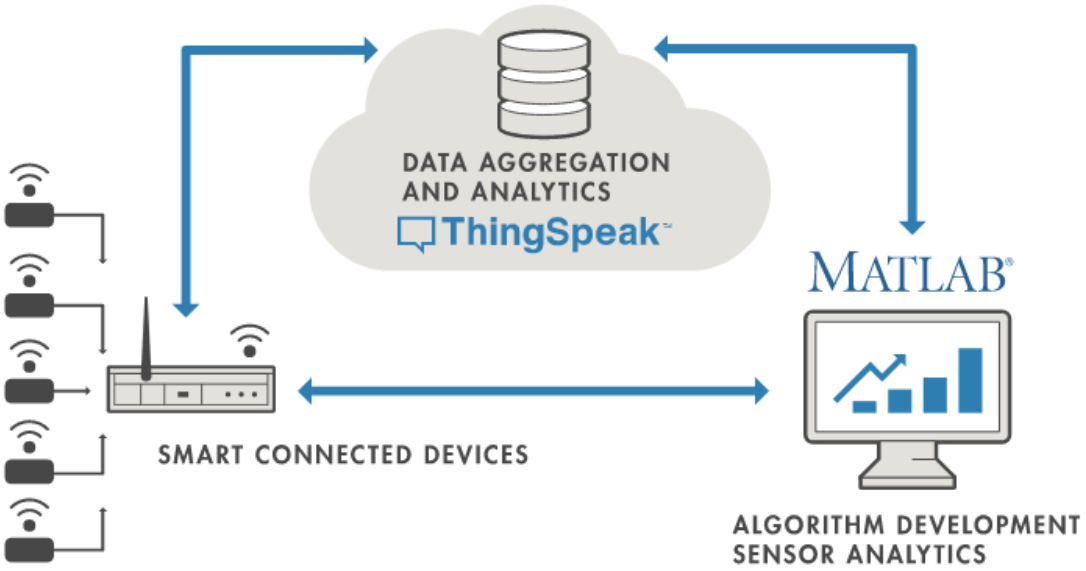


Figure 14 shows IoT System diagram using ThingSpeak

On the left, we have the smart devices (the "things" in IoT) that live at the edge of the network. These devices collect data and include things like wearable devices, wireless temperature sensors, heart rate monitors, and hydraulic pressure sensors, and machines on the factory floor. [22]

In the middle, we have the cloud where data from many sources is aggregated and analyzed in real time, often by an IoT analytics platform designed for this purpose. [22]

The right side of the diagram depicts the algorithm development associated with the IoT application. Here an engineer or data scientist tries to gain insight into the collected data by performing historical analysis on the data. In this case, the data is pulled from the IoT platform into a desktop software environment to enable the engineer or scientist to prototype algorithms that may eventually execute in the cloud or on the smart device itself.

An IoT system includes all these elements. ThingSpeak fits in the cloud part of the diagram and provides a platform to quickly collect and analyze data from internet connected sensors.
More features about ThingSpeak [22]:

- Easily configure devices to send data to ThingSpeak using popular IoT protocols.
- Visualize your sensor data in real-time.
- Aggregate data on-demand from third-party sources.
- Use the power of MATLAB to make sense of your IoT data.
- Run your IoT analytics automatically based on schedules or events.
- Prototype and build IoT systems without setting up servers or developing web software.

6. IFTTT IoT Platform

IFTTT is an **IoT Platform** provide us a free-web based service, helps in connecting different apps and devices with each other and cause automated actions.

IFTTT stands for "If This Then That," where "This" represents a primary app and "That" represents a secondary connected app. When an action takes place on the primary app, the secondary connected app is automatically triggered to also take some sort of action. [23]

we use IFTTT by searching for or creating something called an "applet" (formerly called a recipe). An applet is simply a conditional statement – hence the name If This Then That. Once added or created, the applet creates a chain reaction between at least two of your apps.

IFTTT has a vast library of existing applets created by other users you can use with your own apps. Alternatively, you can create your own applet from scratch.

Some of the most popular apps that can work with IFTTT include [23]:

- Google Calendar
- Amazon Cloud Drive
- Dropbox
- Google Docs
- Google Sheets
- Google Drive
- Facebook
- Messenger
- Skype
- Email
- Gmail
- YouTube

In addition to popular web apps, IFTTT can also connect to certain smart appliances from GE, Samsung, LG, and other brands.

Here in this project, we are using IFTTT on dealing with NodeMCU to send message notifications when the device detects a fall., so let's talk more about it and how it works.

CHAPTER 4

Sensors

In this chapter we are going to focus more on why we chose to use and talk about those specific sensors, in our project we used the temperature sensor, oximeter sensor, and ECG sensor.

We used those 3 sensors as their readings are the main indicator for doctors to predict specific diseases. So, let's take a close look at each sensor and why its reading is important for doctors to know.

Temperature sensor

Measuring body temperature is very important in medicine. A number of diseases are characterized by a change in body temperature. With other illnesses, the course of the disease can be followed by measuring body temperature. This allows the doctor to analyze the effectiveness of treatments based on body temperatures.

A **fever** is the reaction to a disease-specific stimuli. The body changes its normal temperature to support the body's own defense mechanisms. Fever is the most common form of disease-related increase in body temperature. Also, it allows the doctor to analyze the effectiveness of treatments based on body temperatures.

Note: A normal body temperature for younger adult is (36.4-37.6℃) but for the older adult, it is between (35.8-36.9℃). At higher levels (38℃ or more) and lower levels (35℃ or less), asking for a medical help is a must. [24]

Fever Causes

A fever can be a sign of several health conditions, which may or may not need medical treatment.
The most common causes of fever are viral infections as flu and COVID-19. Other causes include [25, 26]:

- Bacterial Infections such as urinary tract infection
- Fungal infections of skin
- Side effects of medications
- Cancer

Fever Diagnosis

Although a fever is easy to measure with a thermometer, finding its cause can be hard. Besides a physical exam, your doctor will ask about symptoms and conditions, medications, and if you've recently traveled to areas with infections or have other infection risks. [26]

Hardware Implementation of Temperature sensor

Description

DS18B20 is a single intelligent temperature sensor, produced by DALLAS semiconductor companies in the United States. It belongs to a new generation of adaptive intelligent temperature sensor and can directly convert temperature signal to the serial digital signals for computer processing. Using strict welding and assembly process, improve the measurement precision, longer service life.

Parameters

Digital chip	DS18B20
Power supply	3.0→5.5VDC
Resolution ratio	9→12 bit
Temperature range	-50→+125℃
Probe size	Diameter: 6→9mm, length:10→1000mm
Probe material	Stainless steel
Connector	Molex, JST, Dupont, CWB, JCT, U type, etc.
Heat-shrinkable sleeve	PVC tube; glass fiber tube; Teflon tube
Connection mode	Black: GND, Yellow: DATA, Red: VDD+

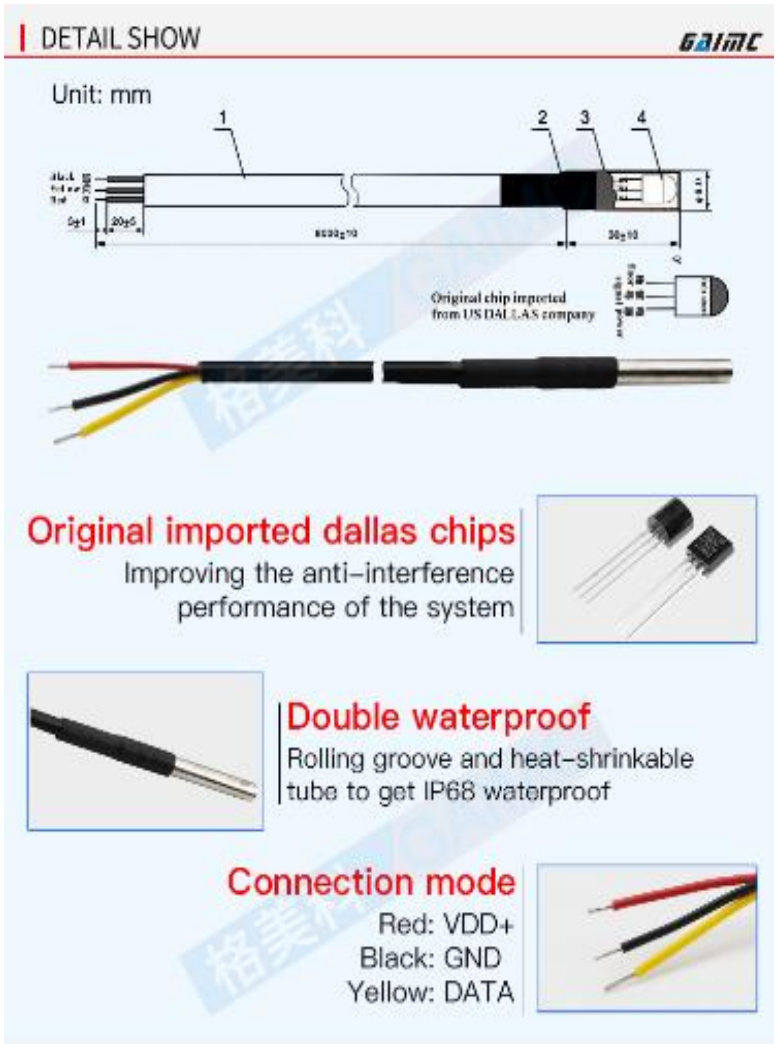


Figure 15 shows Detailed DS18B20 Temperature sensor

Features

- Temperature sensing quickly, Accurate
- Impact resistance, strong waterproof, long service life
- unique single bus interface, multipoint temperature measuring

Connection with Raspberry Pi

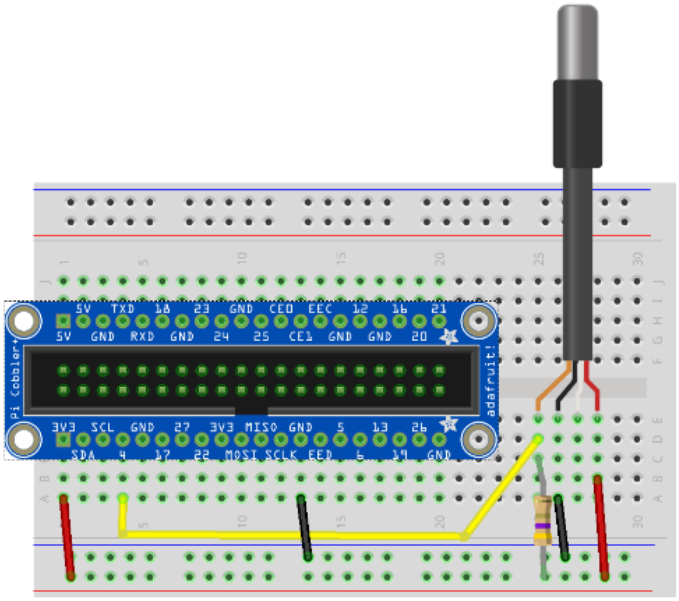


Figure 16 shows DS18B20 connection with Raspberry Pi

Oximeter sensor

The pulse oximeter is a small, clip-like device. It attaches to a body part, most commonly to a finger. Pulse oximetry is a noninvasive test that measures the oxygen saturation level of your blood.

It can rapidly detect even small changes in oxygen levels. These levels show how efficiently blood is carrying oxygen to the extremities furthest from your heart, including your arms and legs. Using pulse oximeter is important as having low oxygen level can be fatal after just a few minutes. [27]

Medical professionals may use pulse oximeters to monitor the health of people with conditions that affect blood oxygen levels, especially while they're in the hospital.

Note: A normal oxygen saturation (SpO2) is (97-100%) but older adult typically has lower level than younger adult (95%). At 92% or less, asking for a medical help is a must. [28]

Low Oxygen causes [28]:

- Anemia
- Heart problems
- High altitude
- Side effects of pain medications
- Asthma: is a condition in which your airways narrow, swell, and may produce extra mucus. This can make breathing difficult and trigger coughing.
- lung damage
- lung disease
- Sleep apnea (stop breathing during sleep)

Doctors use pulse oximetry for several reasons, including:

- to assess how well a new lung medication is working
- to evaluate whether someone needs help breathing
- to evaluate how helpful a ventilator is
- to monitor oxygen levels during or after surgical procedures that require sedation
- to determine whether someone needs supplemental oxygen therapy
- to determine how effective supplemental oxygen therapy is, especially when treatment is new
- to assess someone's ability to tolerate increased physical activity
- to evaluate whether someone momentarily stops breathing while sleeping — like in cases of sleep apnea — during a sleep study

Hardware Implementation of Pulse Oximeter

Description

MAX30102 is an integrated pulse oximetry and heart rate monitor biosensor module. It includes internal LEDs, photodetectors, optical elements, and low-noise electronics with ambient light rejection. The MAX30102 provides a complete system solution to ease the design-in process for mobile and wearable devices. [29]

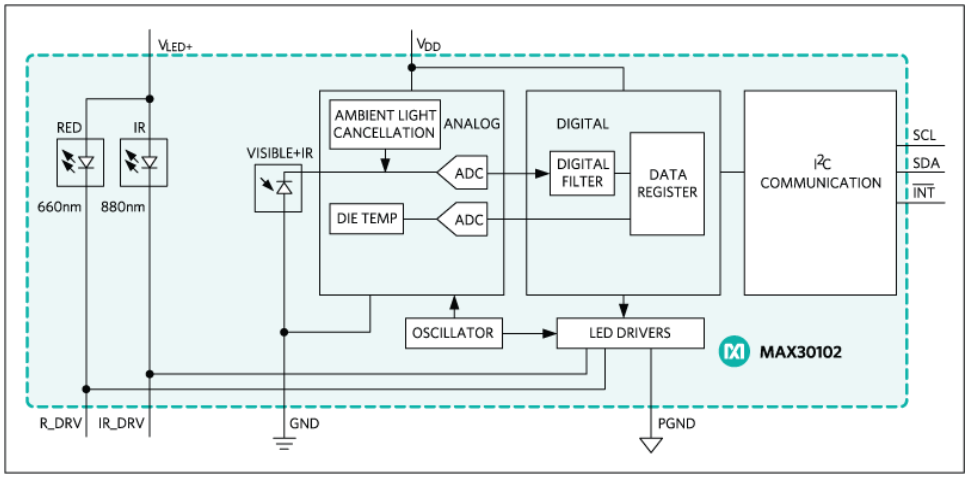


Figure 17 shows MAX30102 Block diagram

Oximetry Sensor IC MAX30102 is a sensor module package whose main component is IC MAX30100. IC MAX30100 is IC which is composed of red and infrared LEDs red and signal conditioners integrated in one IC package. Function from this sensor module it is for retrieve voltage data to search SpO2 value and detect pulses pulse. [30]Block diagram and placement sensor module can be seen in figures 17,18.

Note: A resting heart rate between 60 and 100 beats per minute (BPM) as normal. [31]

Figure 19 shows MAX30102 pinout

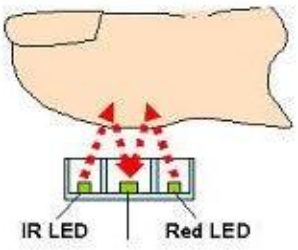


Figure 18 shows MAX30102 placement

Parameters

Component	Characteristics
Power supply	3.3V→5V
Working current	<5mA
Red/IR LED Driving current	0→50mA
Communication	I2C
I2C Addressing	0×57
Operating temperature	-40 ℃→ 85 ℃
Dimension	18×14mm/0.71×55

Features

- Heart rate monitor and pulse oximeter sensor in an LED reflective solution
- Tiny 5.6mm x 3.3mm x 1.55mm 14-pin optical module
- Integrated cover glass for optimal, robust performance
- Ultra-low power operation for mobile devices
- Programmable sample rate and LED current for power saving
- Low-power heart rate monitor (<1mW)
- Ultra-low shutdown current (0.7µA Typ.)
- Fast data output capability
- High sample rates
- Robust motion artifact resilience
- High Signal-to-noise ratio (SNR)

Connection with Raspberry Pi

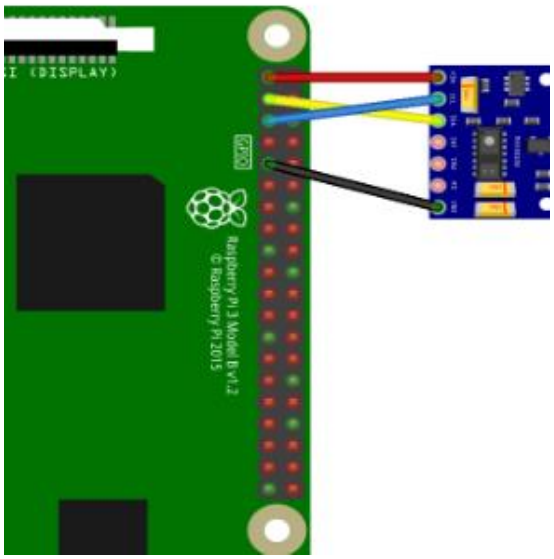


Figure 20 shows MAX30102 connection with Raspberry Pi

Work Principle of Pulse Oximeter

The principles of the sensor operation are very simple: two LEDs, one red (660 nm) and one infrared (880 nm, IR) shine light through human skin. The light is partially absorbed by underlying tissues, including peripheral blood. Sensor's photodetector collects reflected light at both wavelengths and returns two corresponding relative intensities using I2C protocol. [32]

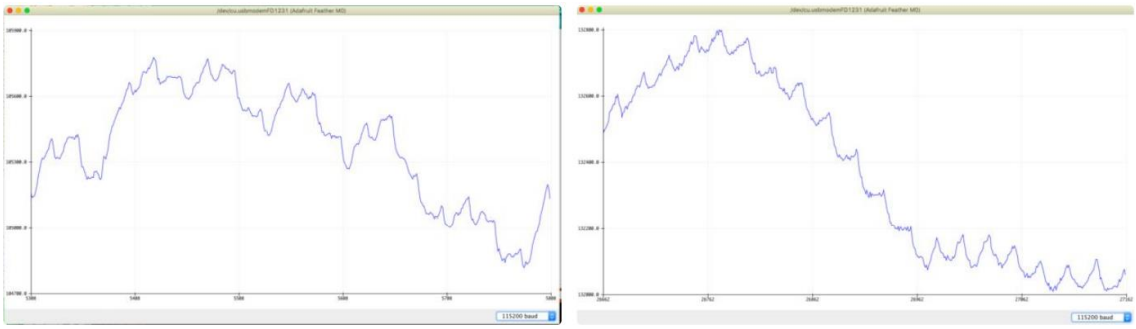


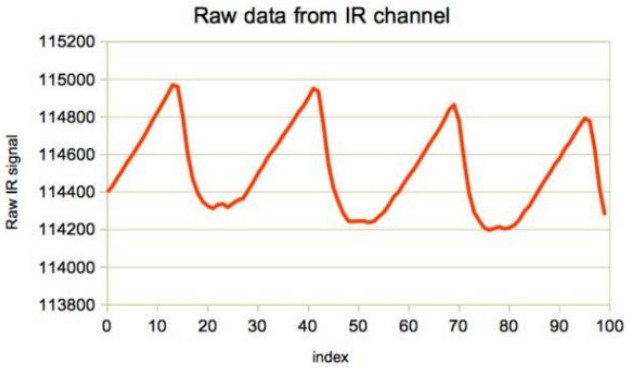
Figure 21 shows IR raw signals

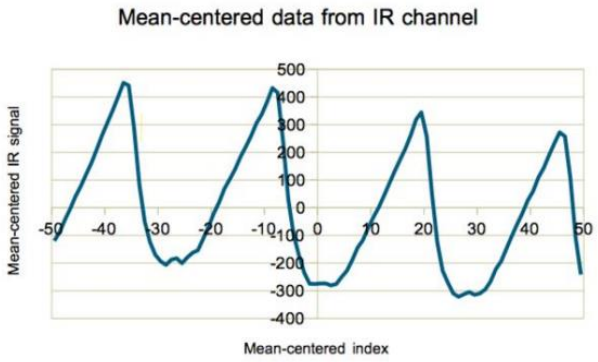
Examples of raw signals (IR channel only) are illustrated in Figure 21. One can notice a periodic component overlaid on a variable baseline that is shifting due to multiple factors mentioned in the Wikipedia page. Motion induced artifacts are particularly annoying since they may mask the useful HR signal and cause bogus results. Hence, advanced commercial oximeters feature accelerometers that help nullify these artifacts. [32]

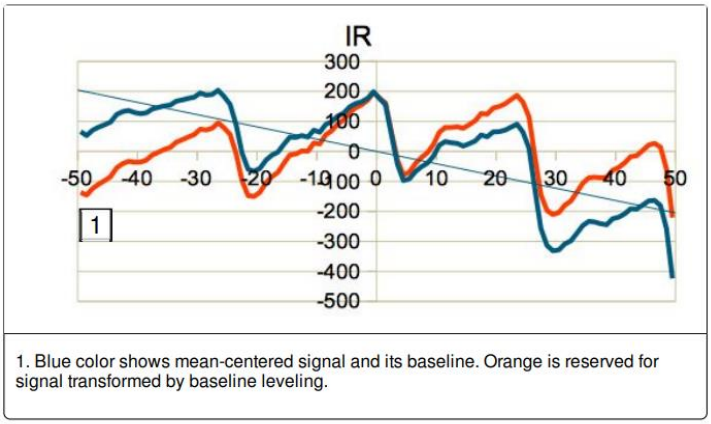
Signal Preprocessing

the raw signal is collected at the rate of 25 H for full 4 seconds, resulting in 100 digitized time points per end data point. Each 100-point sequence must be preprocessed in the following way: [32]

1. Mean-centering; The useful signal, though, is only a part of light reflected from arterial blood which varies on the order of only 10 - first figure.
2. Baseline leveling. Another look at the waveforms shown in Step 2 illustrates that the baseline of real oximetry signals is far from being horizontally flat but varies through different slopes. Third figure shows a mean-centered IR signal (blue curve) and its baseline (blue straight line). In this case, the baseline's slope is negative.
   The signal processing method described ahead requires baseline to be horizontal. This can be achieved by simply subtracting the baseline from the mean-centered signal.







1. Blue color shows mean-centered signal and its baseline. Orange is reserved for signal transformed by baseline leveling.

$$\boldsymbol{\beta} = \frac{\sum_{t=-49.5}^{49.5}(\boldsymbol{t}.\boldsymbol{y}'_t)}{\sum_{t=-49.5}^{49.5}(\boldsymbol{t}^2)} = \frac{\sum_{t=-49.5}^{49.5}(\boldsymbol{t}.\boldsymbol{y}'_t)}{\mathbf{83325}}$$

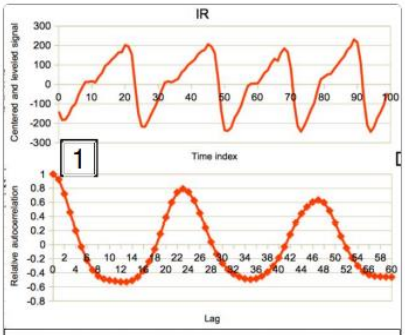
$$\boldsymbol{y_t} = \boldsymbol{y'_t} - \boldsymbol{\beta}.\boldsymbol{t}$$

- $\boldsymbol{\beta}$: slope of the baseline
- $\boldsymbol{t}$: lables time points
- $\boldsymbol{y'_t}$: the intensity of mean centered signal at time point $\boldsymbol{t}$

Autocorrelation Function

the autocorrelation function $r_m$is a quantity found to be very useful in detecting signal's periodicity as well as quality. It is simply a normalized scalar product of the signal's time series with itself shifted by lag m. [32]

$$r_m = \frac{1}{n-m}\sum_{t=1}^{n-m}\left(y_t \cdot y_{(t+m)}\right)$$



1. Good quality signal and its corresponding relative autocorrelation function.

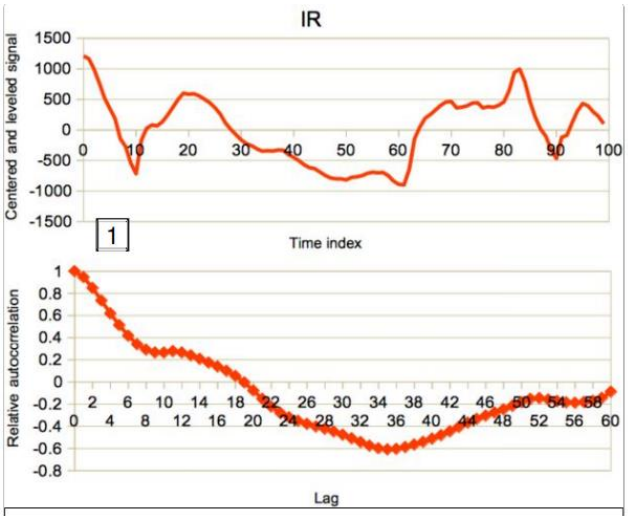
- Plot of the relative autocorrelation of a typical good quality IR signal is shown in the second figure. As expected, it's value at lag = 0 is at its global maximum equal to 1. The next (local) maximum occurs at lag = 23 and equals to 0.79. The presence of local minima and maxima in autocorrelation plot is easy to understand as the signal shifts to the right its peaks interfere destructively with each other at first, but at certain point the interference becomes constructive and achieves maximum at the lag equal higher. then the march continues to the left until $r_{m-1} < r_m$ Thus, determined final m is then returned as the lag at maximum.

- The next iteration starts from that value instead of 25 and the whole process repeats. If the first left neighbor is lower, then the above routine marches lag points to the right in similar manner. In addition, maximum and minimum acceptable lags (corresponding to minimal and maximal heart rate, respectively) are used as limiting values.

- in order to determine the average time period between peaks, from which one can calculate signal's frequency (i.e., heart rate) it is sufficient to find the first local maximum of the autocorrelation function! By default, MAX30102 samples analog input at a rate of 25 points per second, therefore at given m the period in seconds is equal to m / 25.

This leads to heart rate expressed in beats per minute (bpm) by: **HR = 60*25 / m = 1500 / m.**

- algorithm makes the first guess of heart rate = 60 bpm, which corresponds to m = 25. Autocorrelation function is evaluated at that point and compared to the value at its left neighbor, m = 24. If the neighbors value is come out distorted. Such a signal is shown in the third figure. Poor periodicity is reflected in the shape of its autocorrelation function as well as in low value, 0.28, of the first local maximum at m = 11. Compare it to the maximum value of 0.79 determined for the good quality signal. Along with lag limiting values, therefore, the value of r_m / r_o at maximum is a good indicator of signal quality and a requirement for it to exceed certain threshold may be used to filter out motion artifacts. The "RF" graphs shown in the introductions resulted from such threshold equal to 0.25.



1. Distorted signal of poor quality and its corresponding autocorrelation function

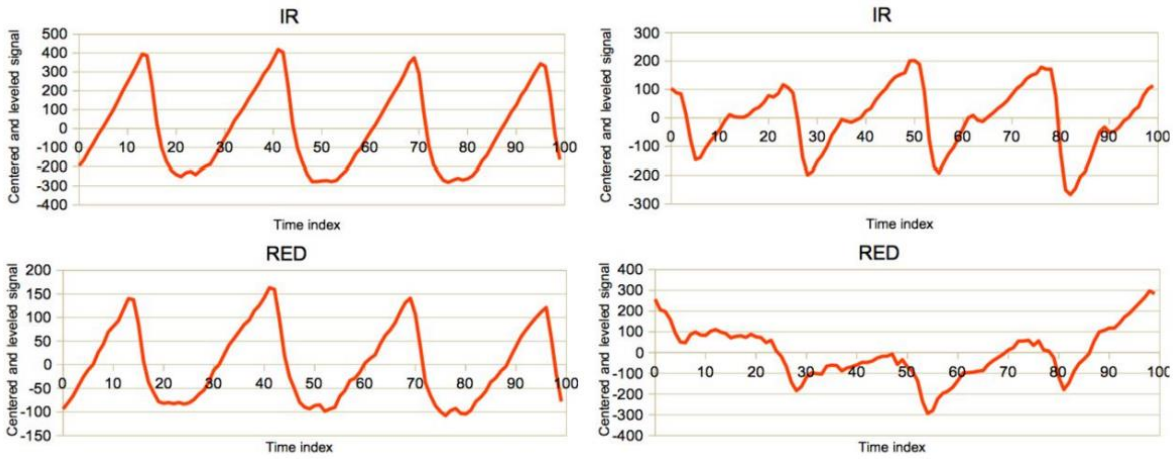
Determining Oxygen Saturation

- To determine SpO2 the red (R) channel must be taken into account. Next, the ratio of red to infrared signals, Z = R/IR, both reflected off the arterial blood.
- How to pick portion of the signal corresponding to arterial blood? Well, this is the pulsatile component that varies with each heartbeat. In words of electrical engineers, it's the "AC part", while the remaining reflected light is the "DC part". Since absolute intensities of R and IR light are not commensurate, the Z ratio is calculated from relative intensities, as shown in the first figure.
- In terms of actually calculated quantities, we use root-mean-square (RMS) of the mean-centered, baseline-leveled signal, y, to the already known mean of the raw signal,
  The nonlinear sensor response requires an empirical calibration between Z and the final SpO2 values. I took the calibration equation from MAXIM's code:
  SpO2 = (-45.06*Z + 30.354)*Z + 94.845

$$Z = \frac{AC_R / DC_R}{AC_{IR} / DC_{IR}} \quad Z = \frac{RMS(y)_R / \langle Y_R \rangle}{RMS(y)_{IR} / \langle Y_{IR} \rangle}$$

- The above procedure still produces a lot of false SpO2 readings. The red channel suffers from many artifacts, just like the IR one. It is reasonable to assume that both signals should be strongly correlated.
  In fact, good quality signals, like the example in third figure, do correlate very well. The Pearson correlation coefficient is in this case as high as 0.99. This is not always the case, as illustrated in the fourth figure. Although the IR signal would pass the heart rate quality filter with its r_m / r_o = 0.76, the distorted R signal results in a poor correlation coefficient between the two equals to only 0.42.

This observation offers the second quality filter: having the correlation coefficient between channels greater than certain threshold.



- In our code the correlation coefficient, cc, is calculated according to the formula in fifth figure, where y represents the mean-centered, baseline-leveled signal.

$$cc = \frac{\sum_{t=-49.5}^{49.5} \left(y_t^R \cdot y_t^{IR}\right)}{\sqrt{r_0^R \cdot r_0^{IR}}}$$

Threshold feature

On having any sensor that its values became bellow or above the normal values mentioned, an E-mail will be sent having the patient's health information to a pre-defined user mail as the shown figure 20.

This is a warning message for a sensor reading as an example, when the BPM measured value was above the normal value (which is 100).

Figure 22 shows A warning Email for crossing a sensor threshold

Electrocardiogram (ECG) sensor

An electrocardiogram (ECG) is a simple test that can be used to check your heart's rhythm and electrical activity. Sensors attached to the skin are used to detect the electrical signals produced by your heart each time it beats. [32]

Our nerve and muscle cells communicate with each other using electrical and chemical signals. Regular electrical signals also control our heartbeat.

These signals are sent by a group of cells in the right atrium of the heart known as the sinoatrial node (SA node), and they spread through the heart muscle tissue as tiny electrical impulses. This causes first the atria and then the ventricles of the heart to contract.

The way that these signals spread through the heart can be measured on the surface of our skin. An ECG measures these changes in electrical signals (in fact, voltage) on different areas of skin and plots them as a graph. The resulting ECG graph is called an electrocardiogram.

These signals are recorded by a machine and are looked at by a doctor to see if they're unusual.

An ECG may be requested by a heart specialist (cardiologist) or any doctor who thinks you might have a problem with your heart.

What does an ECG show?

If the heart is beating steadily, it will produce the typical ECG pattern:

The first peak (P wave) shows how the electrical impulse (excitation) spreads across the two atria of the heart. The atria contract (squeeze), pumping blood into the ventricles, and then immediately relax. The electrical impulse then reaches the ventricles. This can be seen in the Q, R and S waves of the ECG, which is called the QRS complex. The ventricles contract. Then the T wave shows that the electrical impulse has stopped spreading, and the ventricles relax once again. [33]

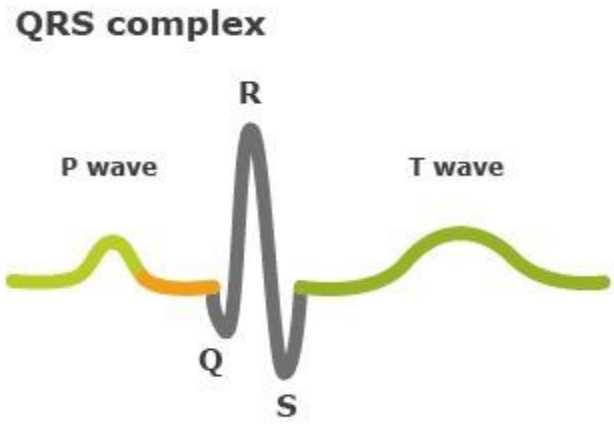


Figure 23 shows ECG wave pattern

An ECG can help detect [32]:

- arrhythmias – where the heart beats too slowly, too quickly, or irregularly
- coronary heart disease – where the heart's blood supply is blocked or interrupted by a build-up of fatty substances
- heart attacks – where the supply of blood to the heart is suddenly blocked
- cardiomyopathy – where the heart walls become thickened or enlarged

In elderly, ECG is very important as the abnormal electrocardiographic changes increase with age and some ECG changes such as T wave abnormalities and first-degree heart block are characteristic of old age. [1]

Cardiovascular diseases remain the leading cause of death of adults over the age of 65. Among these diseases congestive heart failure, coronary artery diseases, hypertension, atrial fibrillation have the greatest significance. [34]

A series of ECGs can also be taken over time to monitor a person already diagnosed with a heart condition or taking medication known to potentially affect the heart.

Types of ECG tests

Resting ECG: This involves lying still on your back with a bare chest. It is important that you lie calmly and comfortably during the test because tensing your muscles, moving, coughing, or shaking can affect the results. The actual measurement takes about one minute, or five minutes at most. [33]

Exercise ECG: Here the electrical activity of your heart is measured while you are physically active. This usually involves riding an exercise bike. The amount of exertion is steadily increased to a high level by making it increasingly difficult to turn the pedals. The test is stopped earlier if any irregularities in the ECG occur. In addition to the ECG graph, this test also provides data on the power that was generated in Watts. Your blood pressure is also checked regularly. [33]

Holter monitor: The electrical activity of the heart is typically recorded over a period of 24 hours. Three or four electrodes are attached to your chest, and a small recording device is worn on a belt or hung around your neck. [33]

The ECG data are then transferred to a computer later on at the doctor's office for analysis. To do this, the doctor also needs information about your daily schedule (like unusual events, physical activity and sleep). A Holter monitor may be used if, for instance, you only have an irregular heartbeat some of the time and it doesn't show up in a “normal” ECG.

Hardware Implementation of ECG

Description

AD8232 ECG Module integrated with AD8232 IC from Analog Devices, which is a single chip designed to extract, amplify, and filter biopotential signals for biopotential measurement applications (like ECG and others). ECGs can be extremely noisy so that the AD8232 Single Lead Heart Rate Monitor acts as an op-amp to help obtain a clear signal from the PR and QT Intervals easily. [35]

It records the electrical activity generated by heart muscle depolarizations (a negative change in the electric charge), which propagates as pulsating electrical waves towards the skin. Although the amount of electricity is in fact very small, it can be picked up reliably with ECG electrodes attached to the skin (in microvolts, or uV).

Figure 24 shows AD8232sensor and its ECG Biomedical connectors

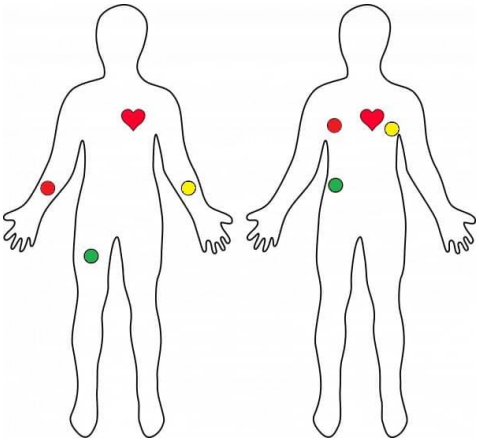


Figure 25 shows ECG Biomedical connectors placement

The full ECG setup comprises at least three electrodes which are placed on the chest or at the extremities as shown on figure 23. [36]

Red : RA (Right Arm)
Yellow : LA (Left Arm)
Green : RL (Right Leg)

ECG electrodes are typically wet sensors, meaning that they require the use of a conductive gel to increase conductivity between skin and electrodes.

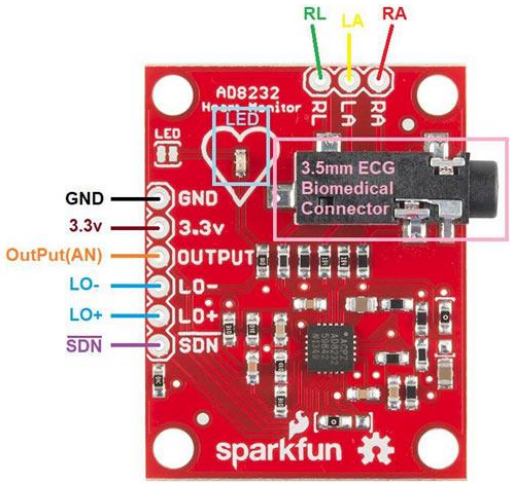


Figure 26 shows AD8232 ECG Module pinout

The AD8232 sensor module is integrated with specially calibrated signal amplifiers and noise filters for ECG signals. As the output of the module is analog type, it is necessary to solder the header pins and connect the module to a microcontroller which has analog input pins like Arduino, ESP32, ESP8266, NodeMCU, or others. [37] But since within our program we are using RPi that does not perform the analog to digital conversion, we used ADS1115 analog to digital converter (ADC) module.

Note: This is not a medically certified device, so it should be used for experiments only and not for clinical use.

Figure 27 shows ADS1115 analog to digital converter module

ADS1115 is a great analog to digital converter that is easy to use with the Raspberry Pi using its I2C communication bus. the ADS1115 is a 16-bit ADC precision with 4 channels. it has a programmable gain from 2/3x to 16x so you can amplify small signals and read them with higher precision. [38]

How do ADC work?

ADCs follow a sequence when converting analog signals to digital. They first sample the signal, then quantify it to determine the resolution of the signal, and finally set binary values and send it to the system to read the digital signal. Two important aspects of the ADC are its sampling rate and resolution. [39]

Sampling Rate (Frequency)

- Tied to the ADC's speed.
- Depends on the type of converter and the needed accuracy
- Measured by samples per second (SPS).

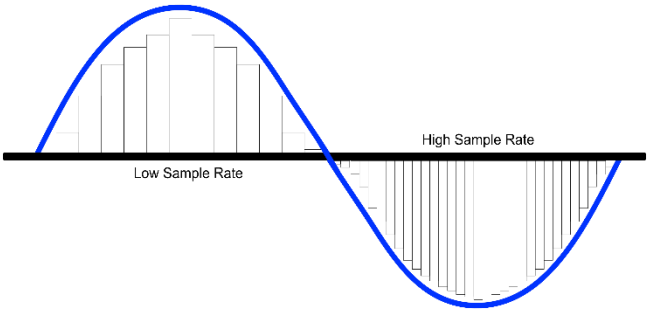


Figure 28 shows ADC Sampling Rate

The equation for sampling rate is:

$$f_s = \frac{1}{T}$$

Where,

f_s, Sample Rate/Frequency

T, Period of the sample or the time it takes before sampling again.

Resolution

- Determined by its bit length.
- The resolution depends on both the bit length and the reference voltage.

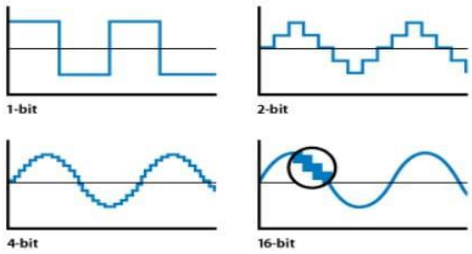


Figure 29 shows the ADC Resolution

These equations help to figure out the total resolution of the signal that you are trying to input in voltage terms:

$$Step\ Size\ = \frac{V_{ref}}{N}$$

Where,

$Step\ Size$, The resolution of each level in terms of voltage

V_{ref} , The voltage reference (range of voltages)

N , Total level size of ADC

To find N size, use this equation:

$$N = 2^n$$

Where, n , Bit Size

The ADS115 executes a conversion rate of about 860 SPS. Also, the device comes with an onboard PGA (programmable gain amplifier). The onboard PGA offers a range input of ±256mV from the PSV (power supply voltage). As a result, the PSV measures large and small high-resolution signals.

Further, the ADS1115 features an input MUX (multiplexer) with four single-ended or two differential inputs. It also works in one trigger mode that powers off automatically after a complete conversion.

ADS1115 Features

- An ultra-small package with a dimension of 2 x 1.5 x 0.4 mm.
- Continuous and single conversion operation mode.
- Two differential inputs
- The device initializes when the power comes up.
- ADS1115's reference input offers negative and positive pressure.
- Digital acquisition output ranges from 0 to 32767 for positive and 32768 to 65535 for negative.
- The conversion rate ranges from 8 to 860Bps.
- Minimal consumption power of 150uA.
- The operating temperature ranges from -40^{0}C to +125^{0}C.

Pin Description of ADS1115 Module

Table 4 Pin Description of the ADS1115 Module

Pin Name	Description [40]
ADDR	I2C address select (slave)
ALERT/RDY	Digital comparator output or conversion ready
GND	Ground
AIN0	Differential channel 1: Single-ended channel 1 input or Negative input
AIN1	Differential channel 1: Single-ended channel 2 input or Negative input
AIN2	Differential channel 2: Single-ended channel 3 input or Positive input
AIN3	Differential channel 2: Single-ended channel 4 input or Negative input
VDD	Power supply: 2.0V to 5.5V
SDA	Serial data: Transmits and receives data (used for I2C communication)
SCL	Serial clock input: Clocks data on SDA (used for I2C communication)

Pin Description of AD8232 ECG Module

Table 5 Pin Description of the AD8232 ECG Module

Pin Name	Description [35]
GND	Power Supply Ground
3.3v	Power Supply 3.3v
Output (ADC)	Operational Amplifier Output. The fully conditioned heart rate signal is present at this output. OUT can be connected to the input of an ADC.
LO-	Leads Off Comparator Output. In dc leads off detection mode, LO− is high when the electrode to −IN is disconnected, and it is low when connected
LO+	Leads Off Comparator Output. In dc leads off detection mode, LOD+ is high when the +IN electrode is disconnected, and it is low when connected

SDN	Shutdown Control Input. Drive SDN low to enter the low power shutdown mode.
RA (Right Arm)	RED Biomedical electrode pad RA (input). Instrumentation Amplifier Negative Input. −IN is typically connected to the right arm (RA) electrode
LA (Left Arm)	YELLOW Biomedical electrode pad LA (input**).** Instrumentation Amplifier Positive Input. +IN is typically connected to the left arm (LA) electrode
RL(Right Leg)	GREEN Biomedical electrode pad RL (input). Right Leg Drive Output. Connect the driven electrode (typically, right leg) to the RL pin.
3.5mm ECG Biomedical Electrode Connector Jack	Combine Biomedical Electrode Pad Connector of RA, LA, RL

Features

- Fully integrated single-lead ECG front end
- Common-mode rejection ratio: 80 dB (dc to 60 Hz)
- Two or three-electrode configurations
- Qualified for automotive application
- Single-supply operation: 2.0 V to 3.5
- Fast restore feature improves filter settling
- Size: 3.5cm x 3cm

Connection with Raspberry pi

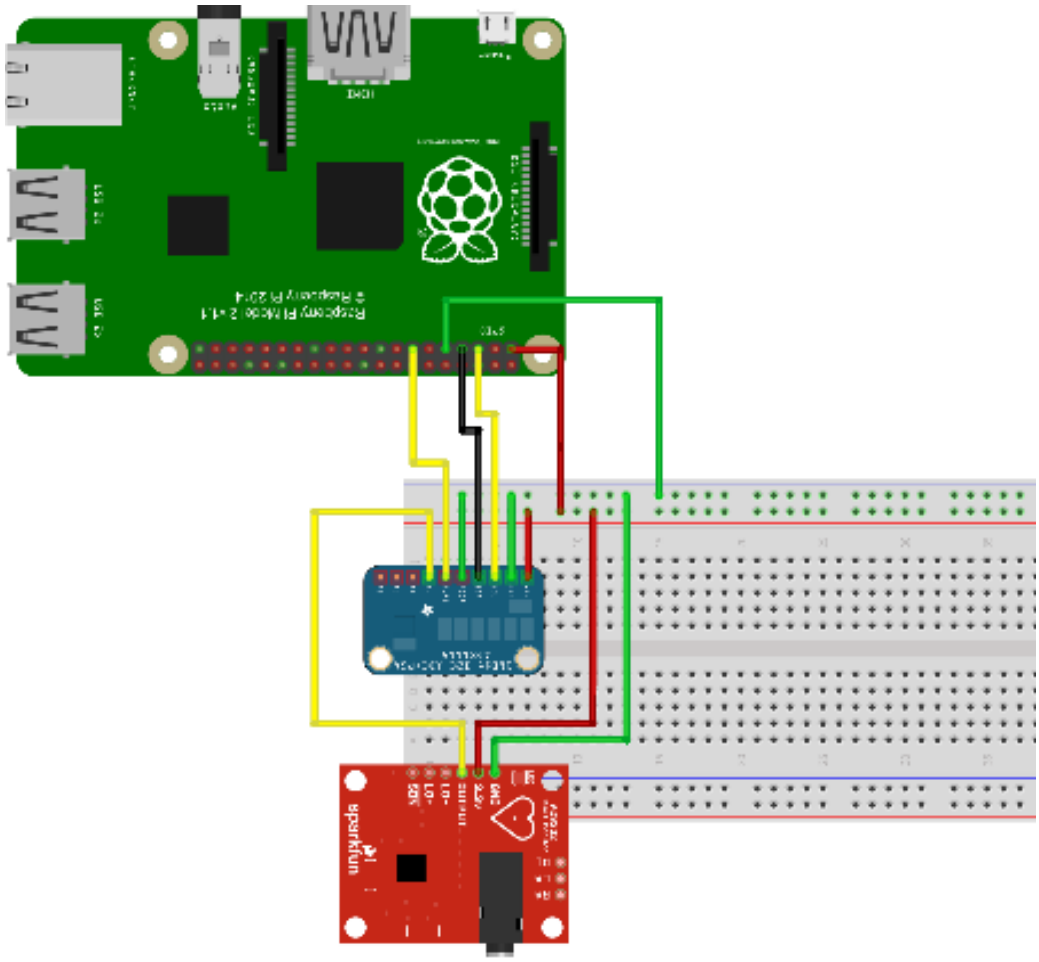


Figure 30 shows AD8232 Connection with Raspberry Pi

*See The RPi sensors' code in Appendix (B).

Fall Detection

Introduction

Falling is an accident that threats the health, especially happened to older people. Caused by reducing levels of strength and stability of the body of a person. Fall detection is very important to monitor someone, especially if the person is elderly. [41]

In 2008, according to N. Noury, more than 33% of people with over 65 years fall each year. Dangers arising from fall like a minor injury, serious injury, dehydration and even death if there is no fast treatment.

Falling is a common problem, but it is quite difficult to define accurately. Since fall is usually characterized by a greater acceleration than the day-to-day activities, the methods are used to measure acceleration usually happens just by using the accelerometer.

Monitoring is necessary for the elderly with a high degree of potential fall. Monitoring can be done by family members or significant others. Surely someone who oversees the elderly should always be near of them so that when the elderly will soon be able to help. But this is it hard for people who care for 24 hours a day. Therefore, monitoring can be done indirectly by utilizing communications technology.

System Design

The architecture of the developed system is as follows. A wearable device is placed on human's waist. The system can detect the elderly's falling by acceleration analysis. Then it will get the elderly's geographic position and send fall alarm short message to caregivers. So, the elderly who has fallen can get timely help to minimize the negative influence. [42]

Fall Detection Algorithm

Overview

Choice of recognition feature has decisive significance to successful fall detection. Information like linear movements (e.g., displacement, velocity, and acceleration) and angular movements (e.g., angle, angular velocity, and angular acceleration) could be obtained directly or indirectly.

Fall detection algorithm design is based on the choice of recognition features. According to the recognition feature, fall detection algorithms are classified as threshold-based and machine learning based. [42]

For threshold-based method, threshold of recognition feature is set by the designer before application which makes the algorithm have rapid response and less resource consumption. But the choice of threshold needs both rigorous schemes and adequate experiments. [42]

For machine learning based design, the classification of fall and normal activities is available with the assistance of technologies such as support vector machine (SVM) and neural network. Machine

learning assistance may enhance system robustness to some extent, but its algorithm design is always high computing resource consumed which limits its application in wearable device.

As the compact wearable device requires low power consumption and a single triaxial accelerometer could provide effective information, threshold-based fall detection algorithm will be used in this system.

Algorithm Design

The proposed system utilized an Inertial Measurement Unit (IMU); which is a sensor used to understand the orientation of a body in three-dimensional space and can provide information on the body's angular rate, orientation, and even force applied to it by measuring the linear and angular motion using the combination of gyroscope, accelerometer, and magnetometer. [43]

→ *Gyroscope*

A gyroscope (sometimes simply called a gyro) measures the rate of rotation and tracks it over time to calculate the current angle. As it tracks the rate of rotation, the gyro drifts. Gyros work well for measuring quick, sharp movements.

→ *Accelerometers*

Accelerometers are real-time compared to gyros as they don't have to be tracked all the time, they can give the angle values at any moment. They are useful in measuring both gravity-based static movements as well as more dynamic movements based on inertia and acceleration. The downside is that they can get noisy, so they need to be used for measuring angles over a period of time.

→ *Magnetometer*

A magnetometer uses the earth's magnetic field to understand the direction. In IMUs they are mostly used in combination with an accelerometer and a gyro in order to compensate for the long-term drift in sensor data from the gyro and to correct the gyro bias, but we will not use it in our system.

The Algorithm used in this fall alarm system is based on thresholds of sum acceleration and rotation angle information. When a real fall happens, collision between human's body and ground will produce obvious peak value at the sum acceleration $|a|$which has magnitude as [42]

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

where (a_x), (a_y) and (a_z)present accelerometer measurements of three axes. The system uses the sum acceleration as the first step to distinguish high intensity movements from others. But normal motions such as jumping or sitting also produce peak values, which mean that additional detection features are required.

The second feature used here is an angle calculated based on acceleration measurements. As human's motion has low acceleration, it is feasible to get gravity component in each axis by using a low pass filter. If gravity components could be separated before and after human's fall, then it is possible to

calculate the rotation angle of accelerometer coordinate in 3D space, which is also equivalent to the rotation angle of gravity vector relative to fixed coordinate. Coordinate constructed by the accelerometer and the gravity vector is shown in Figures 29,30.

the gravity vector g which could be described as:

$$|g| = \sqrt{g_x^2 + g_y^2 + g_z^2}$$

While its three components are denoted as (g_x), (g_y) and (g_z).

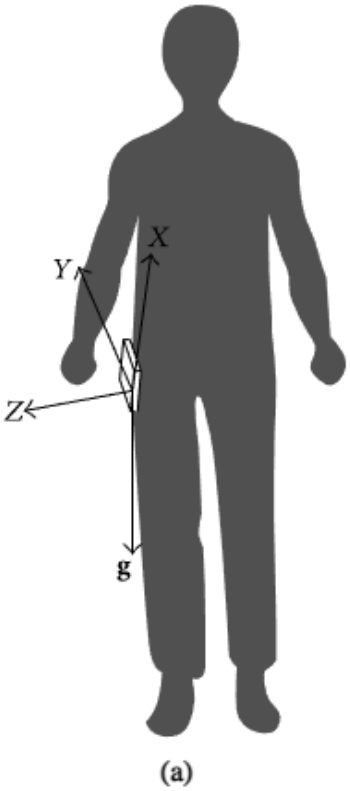


Figure 32 shows Coordinate and gravity before falling

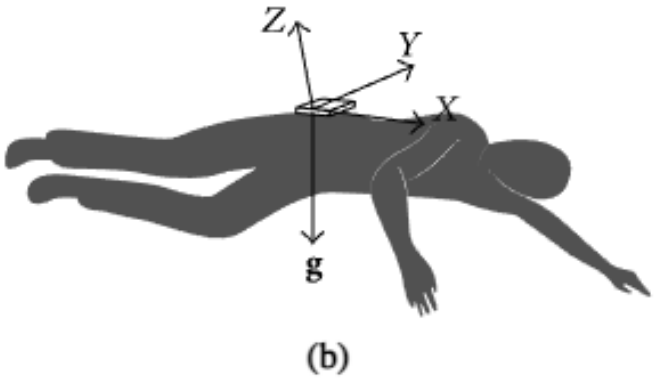


Figure 31 shows Coordinate and gravity after falling

What's referred to as the *Proof Mass* is suspended on springs and is free to move around as the device undergoes acceleration. The fixed comb of electrodes sets up a capacitive effect between itself and the proof mass. As the device moves a change in capacitance is recorded and converted by an ADC to a digital value between 0 and 32,750. The gyroscope functions in a similar way except that it works on the *Coriolis Effect* instead of acceleration. [44]

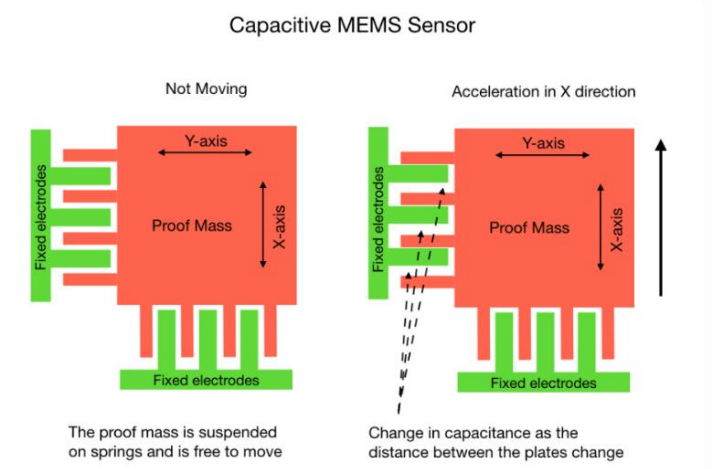


Figure 33 shows Capacitive MEMS Sensor

Device Sensitivity

The analog voltage read from the capacitive sensors is converted to digital signal in the range of 0 to 32750 values. These values make up the measurement units for the gyroscope and accelerometer. The measurement units have to be split up to represent meaningful information. [44]

For most applications we can set the sensitivity level to 250°, which is the default setting. This gives us 131 measurement units per second per degree providing a very high level of precision. [44]

The default setting for the accelerometer is 2g. This should be appropriate for most applications

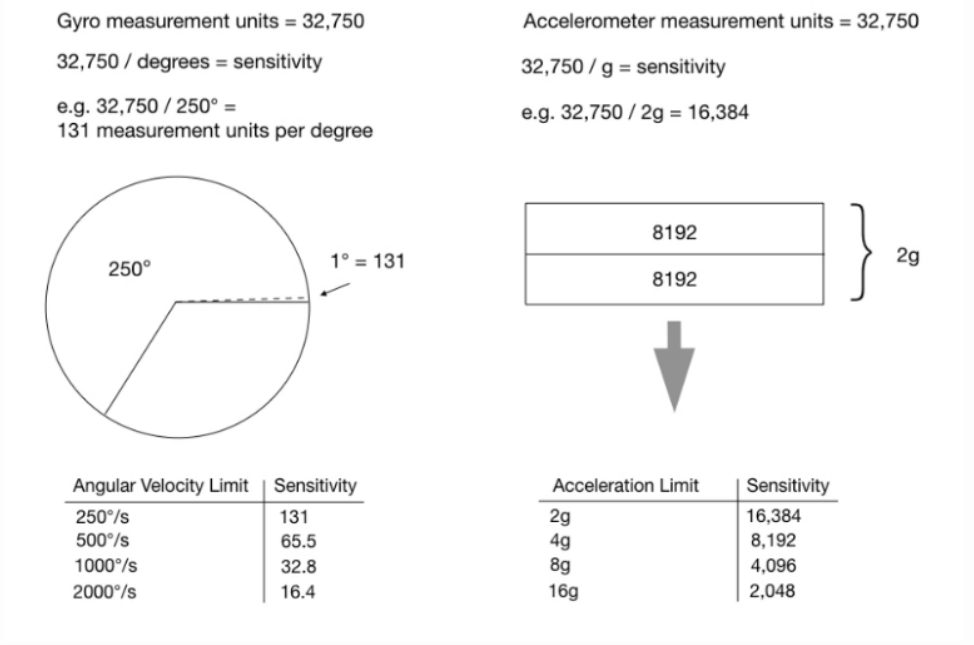


Figure 34 shows Sensitivity Measurements

Note: The Fall Detection sensor code in Appendix (C), we subtract the threshold values from the detected values of acceleration and gravity components to get values relative to the falling position.

The wearable device will be mounted on human's waist at first to reflect the motion of human body closely, and the device will record high $|g|$ values while the elderly is standing still. There is no special requirement of the device orientation but only keep stationary during the wear.

When measurements in three different axes have been acquired, the sum acceleration $|a|$ will be calculated. When a real fall happens, sum acceleration will reach peak value ($0 \leq |a| \leq 10$). In a real falling, the variation of acceleration will stop for a time duration (500 millisecond) as the elderly will lie on the ground. [42]

Hardware Implementation of Fall Detection Sensor

The circuit is built around NodeMCU ESP8266 and an MPU9250 accelerometer and gyro breakout module. NodeMCU is used here as a microcontroller and Wi-Fi module to connect with IFTTT to send Short Message Service (SMS).

MPU9250 is a multi-tasking sensor module based on MEMS (Micro Electro-Mechanical Systems) architecture. It is a product of Microdevices Corporation. The MPU9250 is a 9-axial Motion Tracking device with a 3-Dimensional gyroscope, 3-Dimensional magnetometer, and 3-Dimensional accelerometer embedded into an ultra-compact IC. The high-performance IC has 16-bit resolution analog-to-digital Converters. The sensitive module also has an inbuilt temperature sensor and an auxiliary I2C interface data transmission with non-inertial sensors. [45]

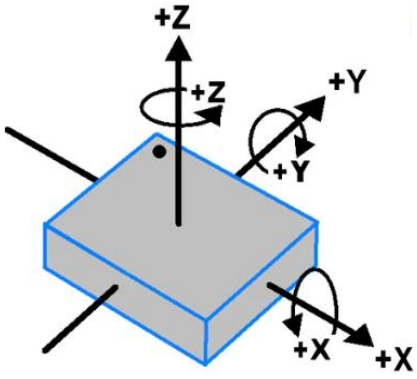


Figure 35 shows MPU9250 Orientation and polarity of rotation

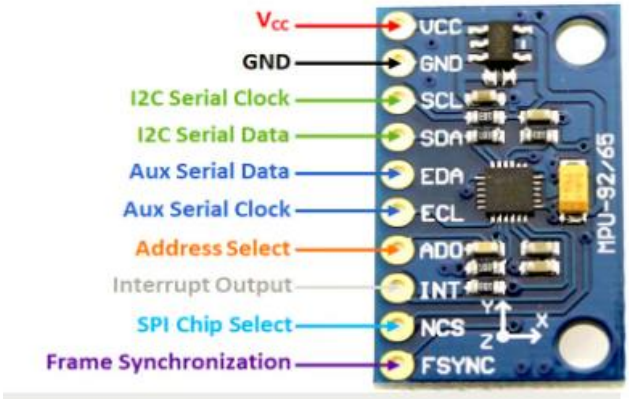


Figure 36 shows MPU9250 Pinout

MPU9250 Pinout Configuration

Table 6 MPU9250 Pinout Configuration

Pin Number	Pin Name	Description [46]
1	V_{CC}	Power Supply 3.0 – 5.0 volts
2	GND	Ground Reference
3	SCL	I2C Serial Clock
4	SDA	I2C Serial Data
5	EDA	Auxiliary Serial Data used for off-chip sensor communication.
6	ECL	Auxiliary Serial Clock
7	AD0	I2C/SPI Address Select
8	INT	Interrupt: FIFO (First in, first out) Overflow, data ready and wake-on-motion (acceleration) interrupt
9	NCS	SPI Chip Select
10	FSYNC	Frame Synchronization used to synchronize data frames in case of data transmission errors.

MPU9250 Features

- Supply Voltage – 5V (typical)
- Supply Current – 4mA (typical)
- Accelerometer Range - ±2g, ±4g, ±8g, ±16g
- Gyroscope Range - ±250°/second, ±500°/second, ±1000°/second, ±2000°/second
- Magnetometer range - ±4800uT
- I2C, SPI, and Auxiliary I2C [46]

Connection with NodeMCU

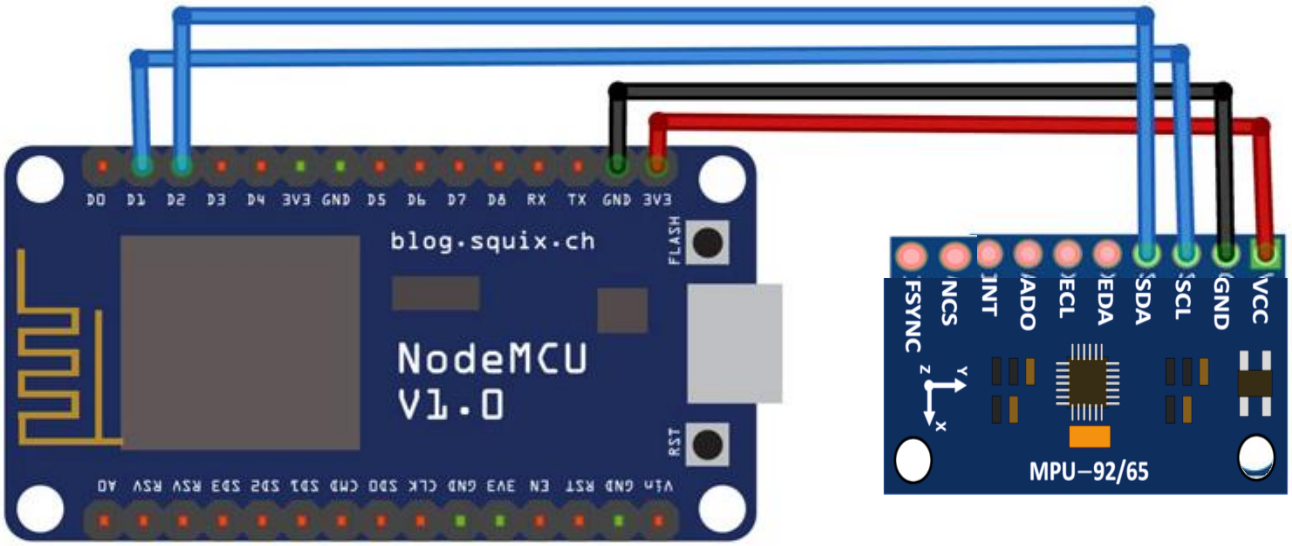


Figure 37 shows MPU9250 Connection with NodeMCU

CHAPTER 4

Cloud Platforms Connection

As mentioned in the chapter related to the various types of clouds, we specifically chose ThingSpeak and IFTTT to be used in our project. So, let's get close on their step-up steps and usage for our project.

ThingSpeak

ThingSpeak provides various services exclusively targeted for building IoT applications. It offers the capabilities of real-time data collection, visualizing the collected data in the form of charts, ability to create plugins and apps for collaborating with web services, social network, and other APIs. [47]

Therefore, we used it with RPi collecting the data of Temperature, Oximeter, and ECG sensors.

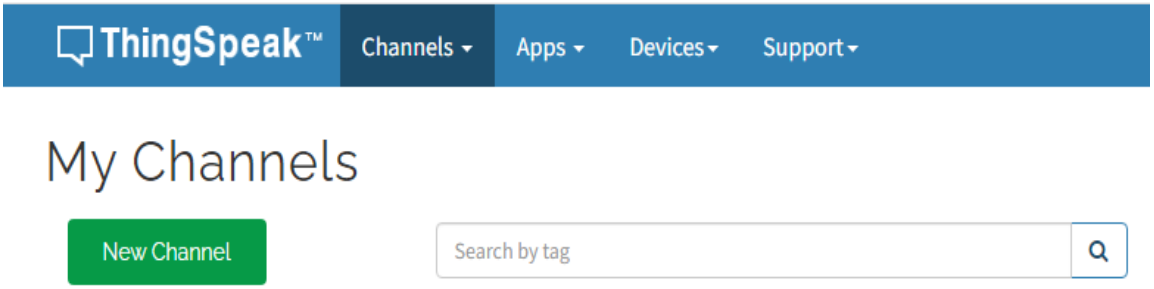
The core element of ThingSpeak is a 'ThingSpeak Channel'. A channel stores the data that we send to ThingSpeak and comprises of the below elements [47]:

- 8 fields for storing data of any type - These can be used to store the data from a sensor or from an embedded device.
- 3 location fields - Can be used to store the latitude, longitude, and the elevation. These are very useful for tracking a moving device.
- 1 status field - A short message to describe the data stored in the channel.

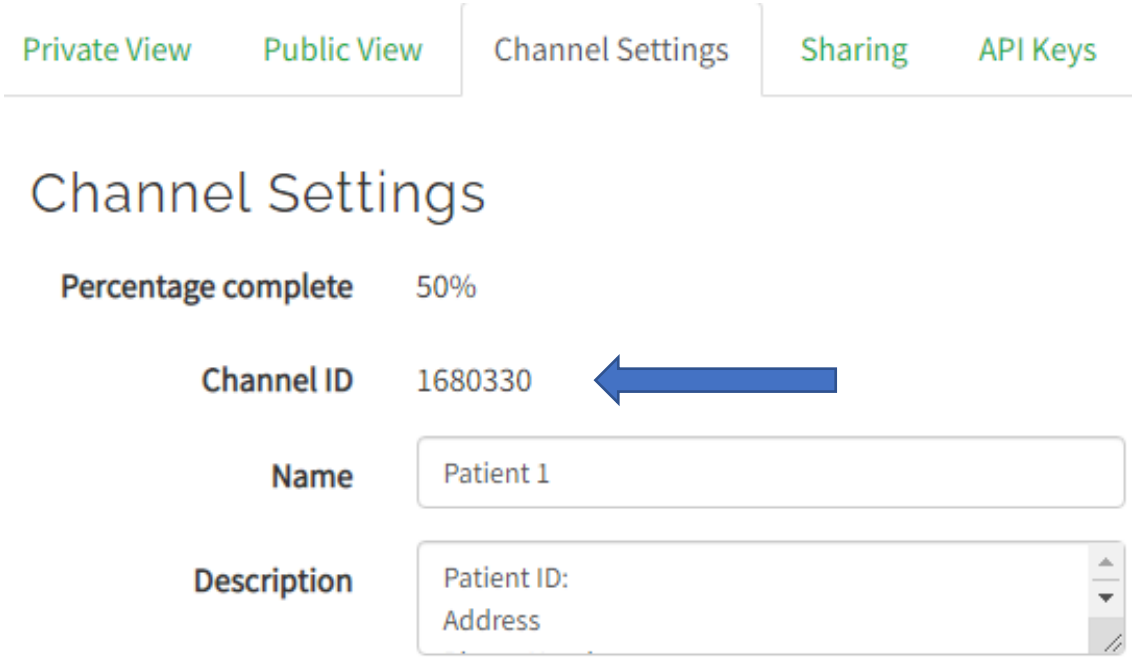
To use ThingSpeak, we need to sign up and create a channel. Once we have a channel, we can send the data, allow ThingSpeak to process it and also retrieve the same. Let us start exploring ThingSpeak by signing up and setting up a channel.

Steps of connecting our project to ThingSpeak cloud

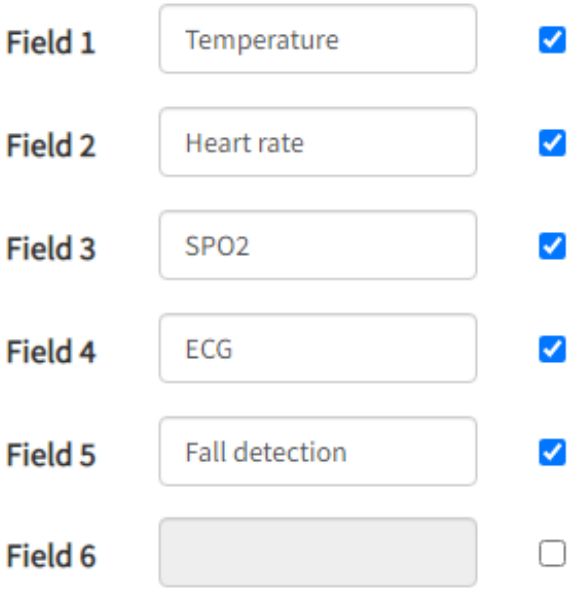
1. First sign up in ThingSpeak main page with your own e-mail address and password you create. [48]
2. Then create new channel.



3. Name your project from channel settings.

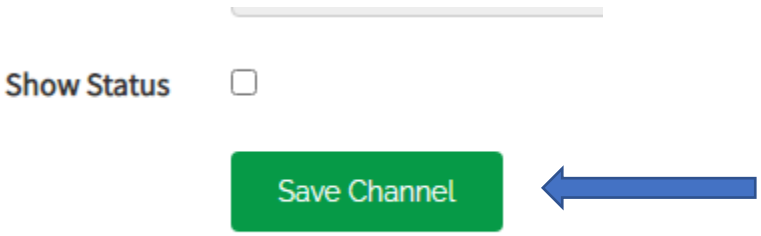


4. From channel settings as well, you can select number of fields depends on number of used sensors, in our project we used temperature, oximeter, ECG, and fall detection sensor so we applied five fields as shown in fig below.

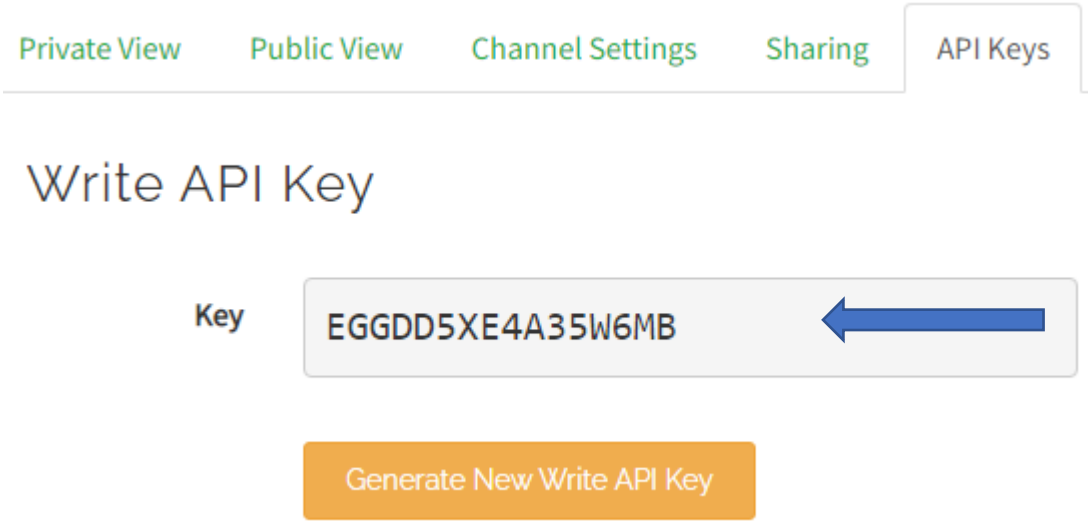


We can give a name to our fields as well depending on the sensors used and give check mark beside each used field.

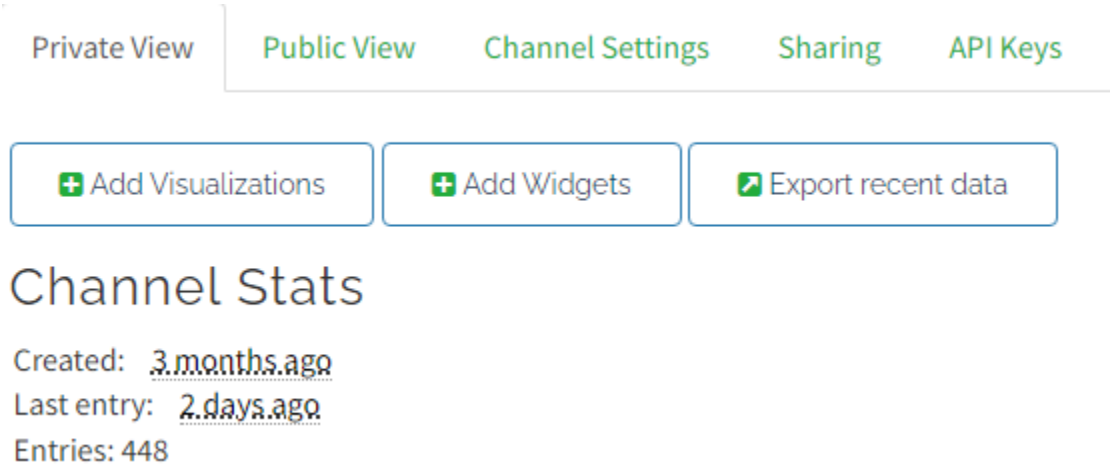
5. Then click on "Save channel" after completing channel settings.



6. Get an API key.



7. Here you have two options to choose whether public view or private view.

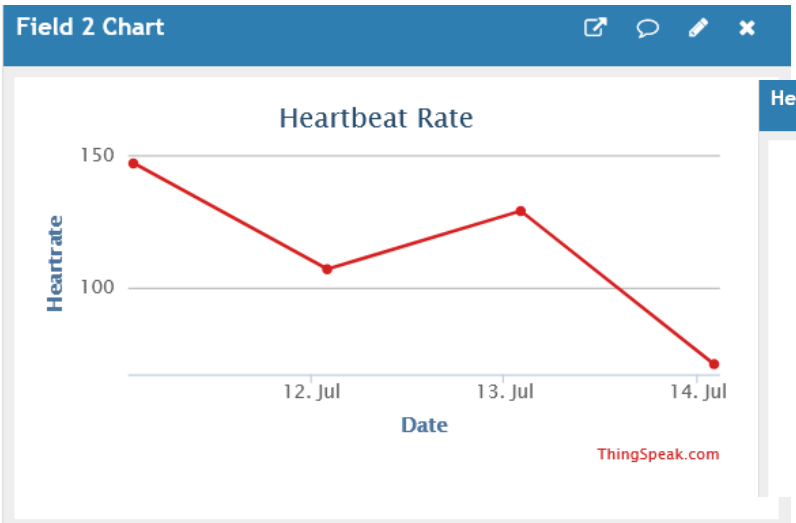


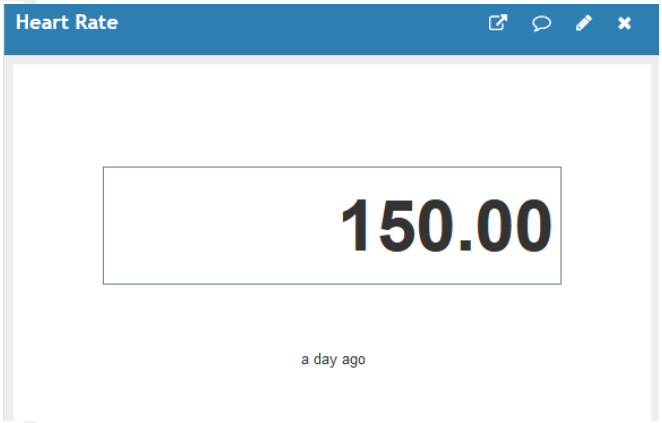
- In private view all the sensor readings and charts will appear there.

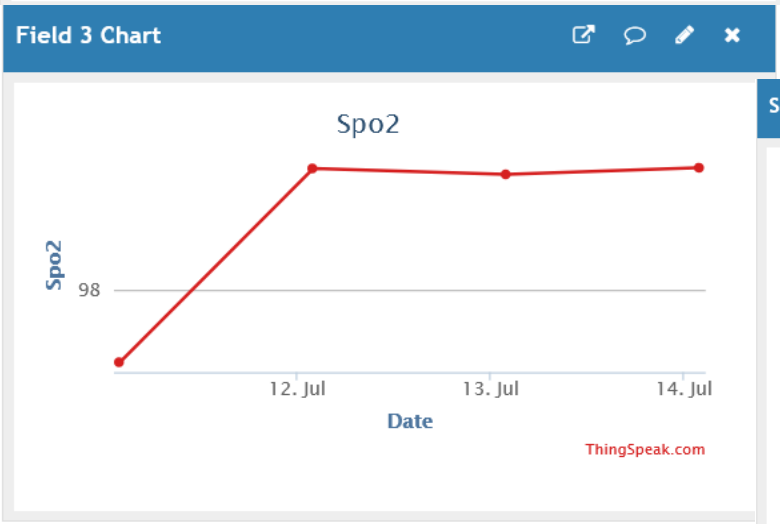
- To make channel public go to 'sharing' as shown in fig above.

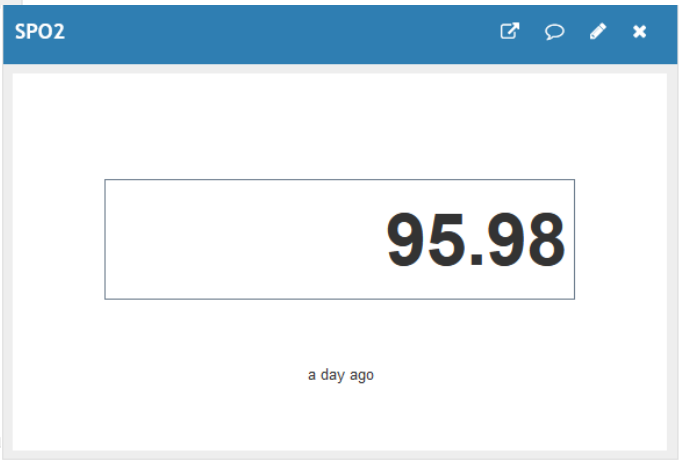
The following figures show the project channels:

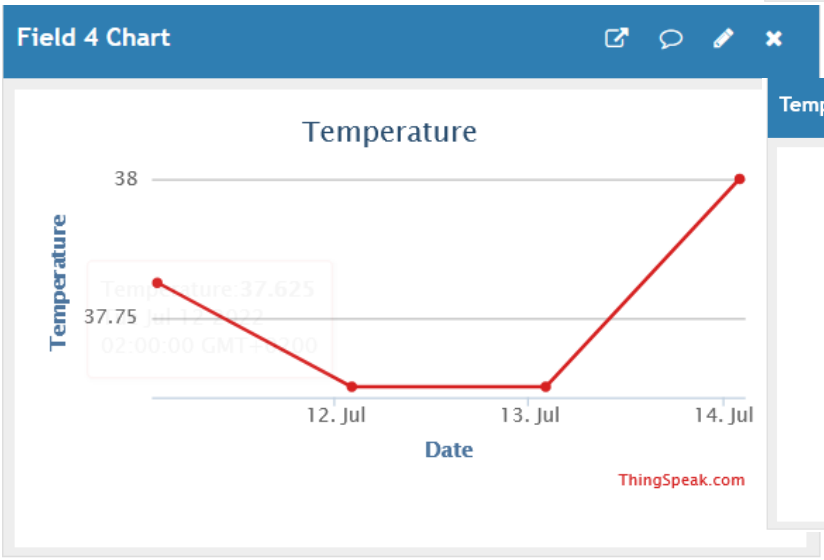
* This graph is drawn by MATLAB on ThingSpeak from The ECG_Readings.csv created carrying the ECG voltage. See Appendix (E).

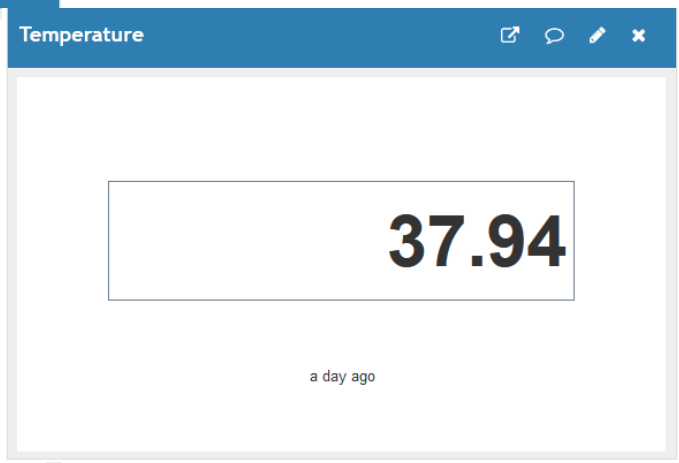

Field 2 Chart
Heartbeat Rate
150
100
Heartrate
12. Jul
13. Jul
14. Jul
Date
ThingSpeak.com


Heart Rate
150.00
a day ago


Field 3 Chart
Spo2
98
Spo2
12. Jul
13. Jul
14. Jul
Date
ThingSpeak.com


SPO2
95.98
a day ago

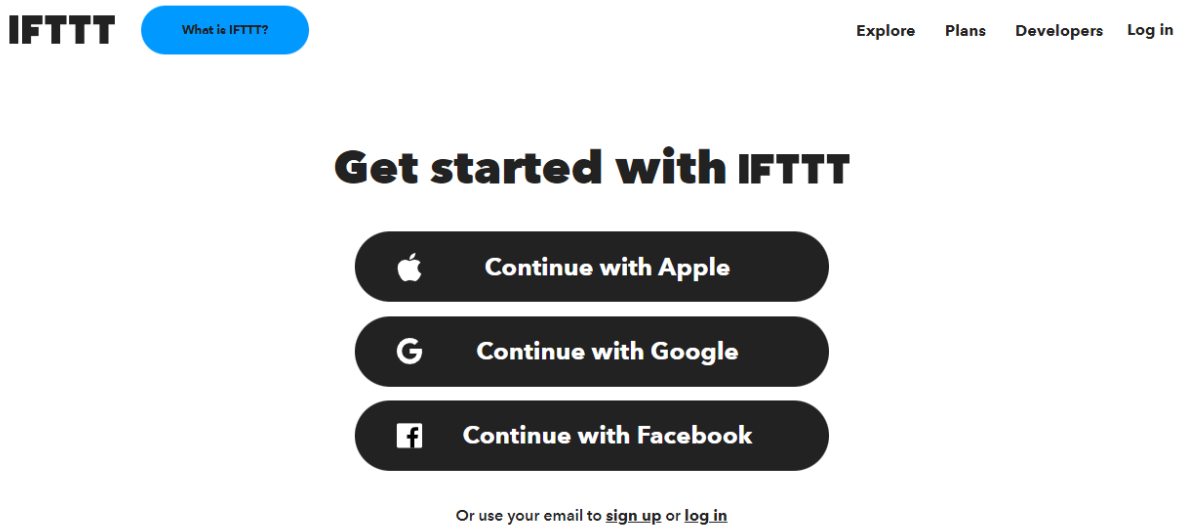

Field 4 Chart
Temperature
38
37.75
Temperature
12. Jul
13. Jul
14. Jul
Date
ThingSpeak.com


Temperature
37.94
a day ago

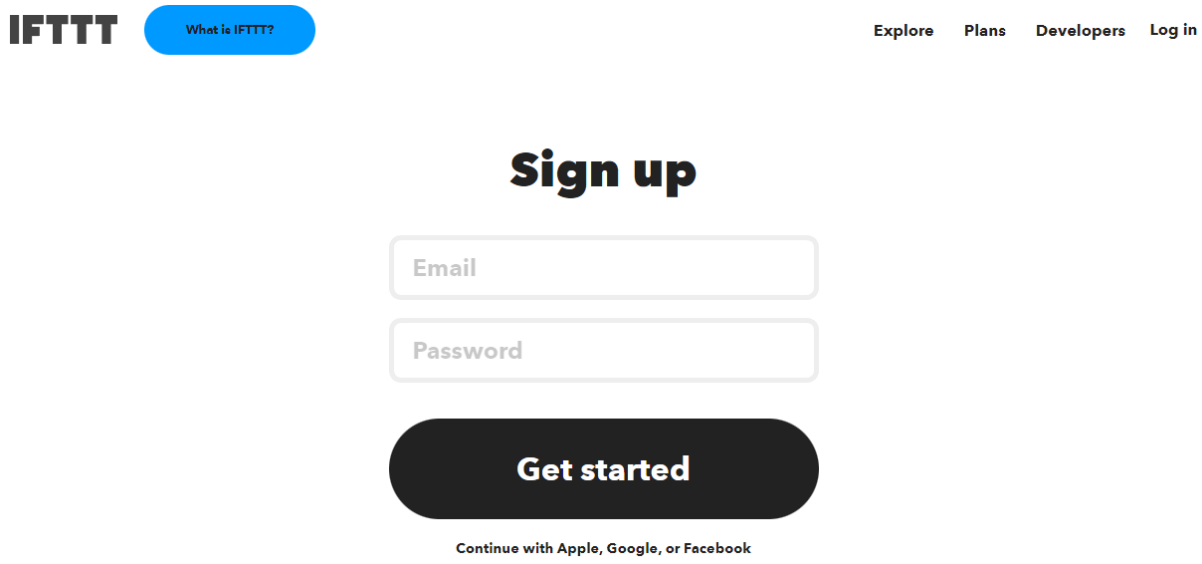
IFTTT

IFTTT Setup for RPi Sensors

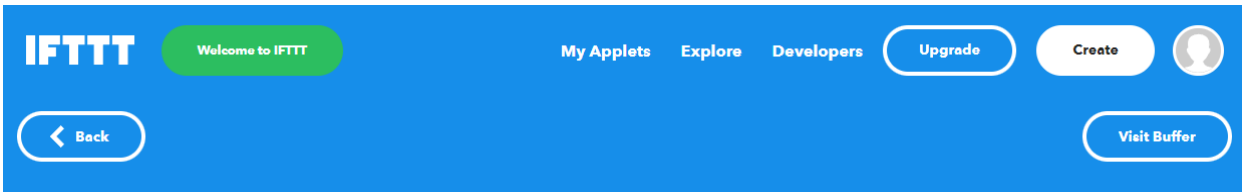
1. To use the IFTTT, sign in to your IFTTT account if you already have one or create an account. [49, 50]



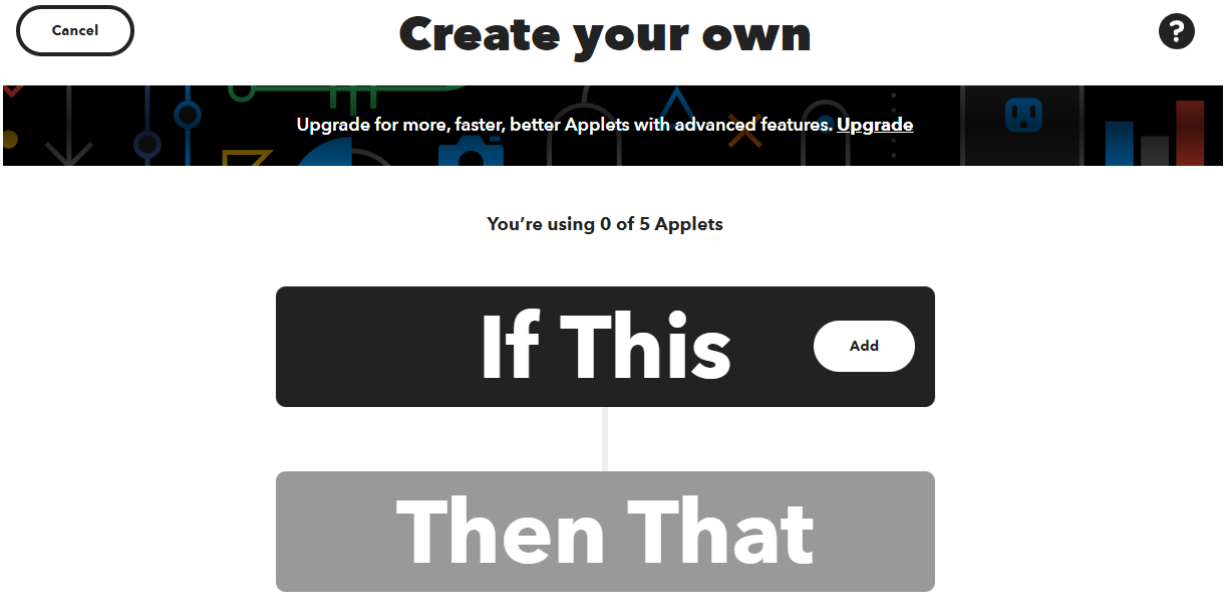
2. Click the ‘sign up’ tag. You will see the following window pop up. Enter your email address and password to start working in IFTTT. This whole process is free of cost for the first five applets.



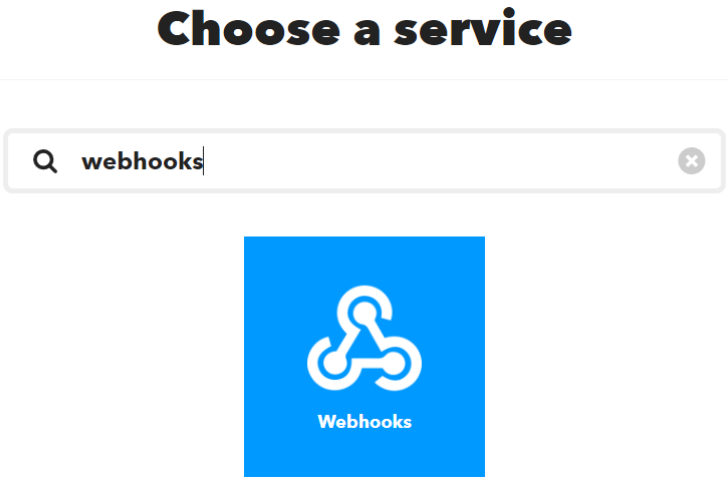
3. After you have created your account, we will be directed to the page where we will create our applet. Click on ‘Create.



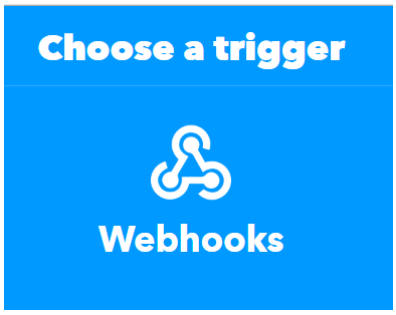
4. The following window opens. Click the following Add button in the ‘If This’ section.

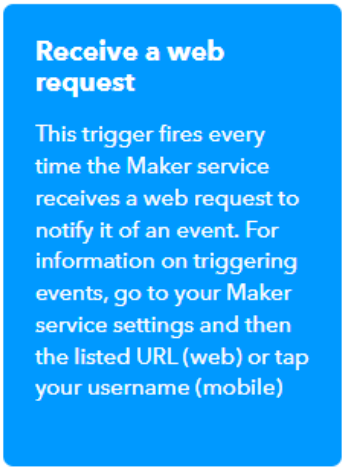


5. Another page will open in which we will have to choose our service. There is a lot of options to choose from. Write down ‘webhooks’ in the search option and its icon will appear:

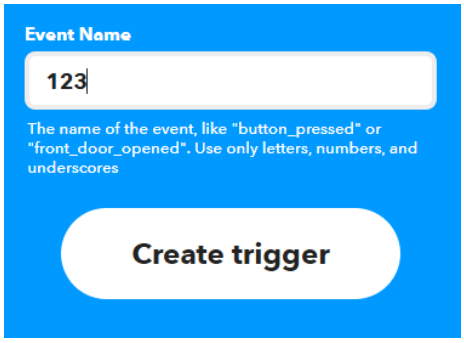


6. Next, choose the trigger as: ‘Receive a web request’ by clicking on it. Whenever webhooks will receive a web request, some action will take place. This we will define in the ‘THAT’ section.

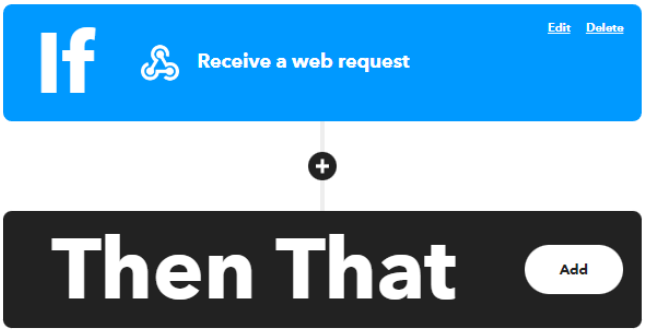




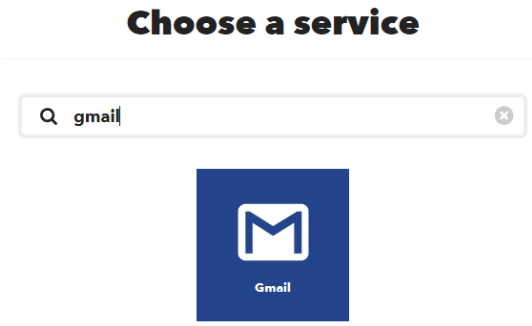
7. After clicking the Receive a web request, the following window will open up. We will write down ‘123’ as the event name for the web request – ‘123’ is the Egyptian number for emergency medical care -. Click ‘Create Trigger’ button.



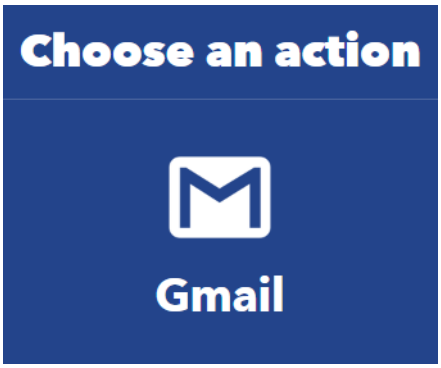
8. After the trigger is created, we are taken back to the web page where we first added the service for the ‘IF THIS’ section. Now we will click the ADD button for the ‘THEN THAT’ section.

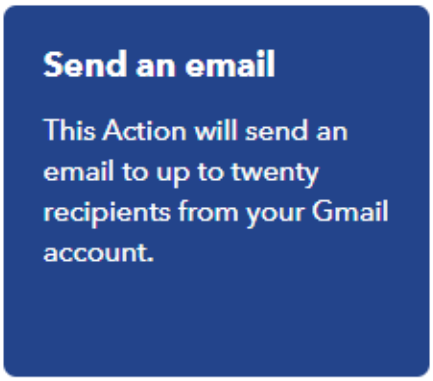


9. Now we will choose the service. We must choose what will happen if a web request is received. We will type ‘Gmail’ in the search option and click on its icon. This is because we want to react ‘***123’*** event by sending a message.

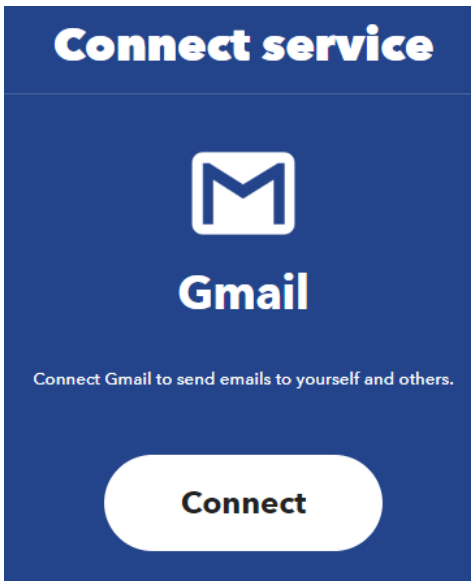


10. The following page opens. Choose ‘Send an email’ to proceed further.



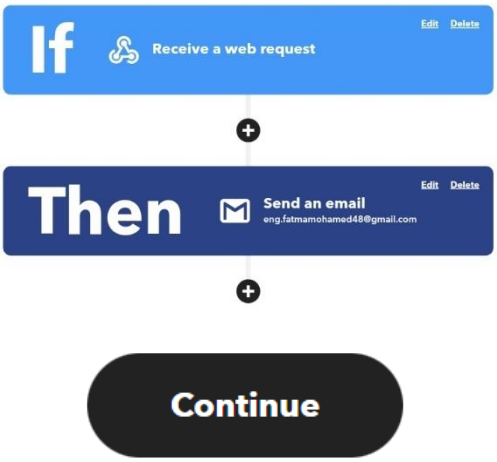


11. Connect the project mail to the Gmail service.

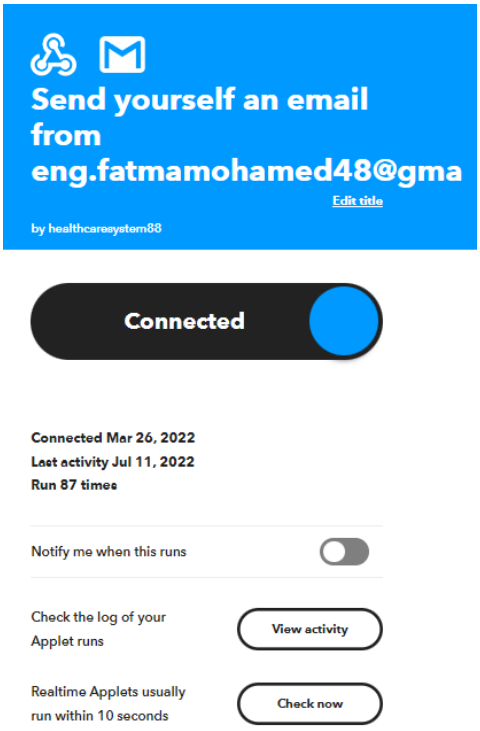


12. Now, it will ask you to enter the phone number and message body. Enter the Mobile Number and the message body and then click on '***create action'*** to complete the process.

13. After we have created the action, we will be guided towards the initial web page of IFTTT. Click 'Continue' to proceed.

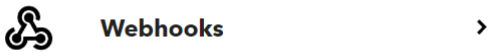


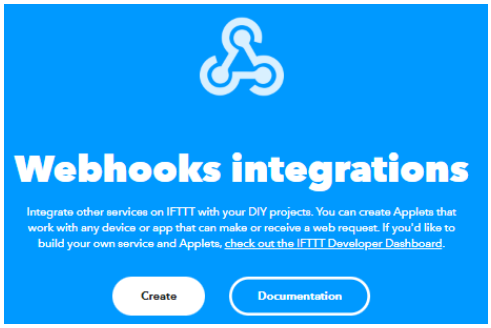
14. After this click the Finish button. Make sure to turn ON the notifications when the applet is running. You have successfully created the applet as shown below.



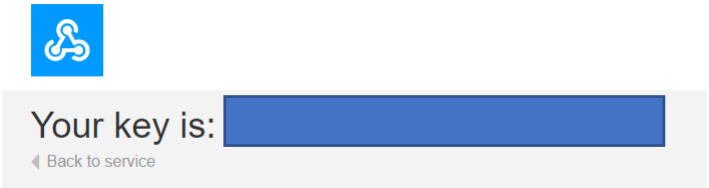
15. Before we proceed further with our project, we want to access our private key. This is important as it will be required while programming.

    Go to your applet and select "My Services" The following windows will appear. Afterward, click on Webhooks.





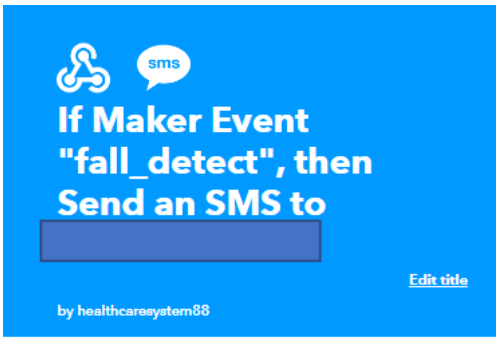
16. This will take you to the following web page. Click on 'Documentation.'
17. You will receive a key that will be used in programming.

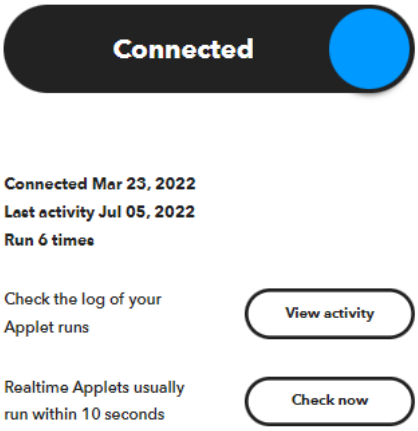


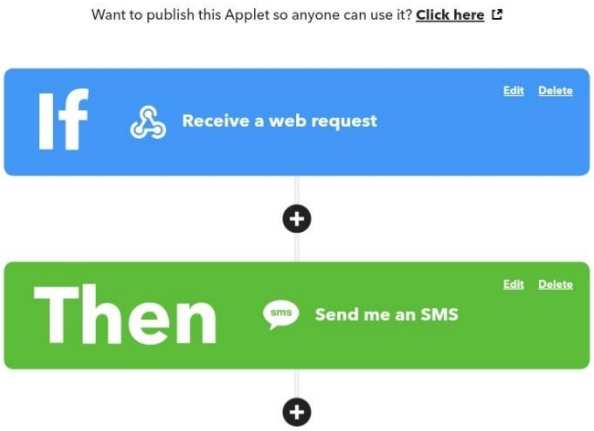
*See The RPi sensors' code in Appendix (B).

IFTTT Setup for Fall Detection Sensor

1. Do as the previous steps but with using ‘SMS’ service instead of ‘Gmail’. [49,







50]

*See The Fall Detection sensor code in Appendix (C).

CHAPTER 5

Mobile Application

Introduction

As we discussed, ThingSpeak has APIs for collecting data produced by sensors and APIs for reading that data from applications. Think of an IoT project as two parts. One part of the project is where you need to program a thing to send data. And the second part is where you want to see that data. ThingSpeak sits in the middle and makes it handy to do both as shown in figure 34. [51]

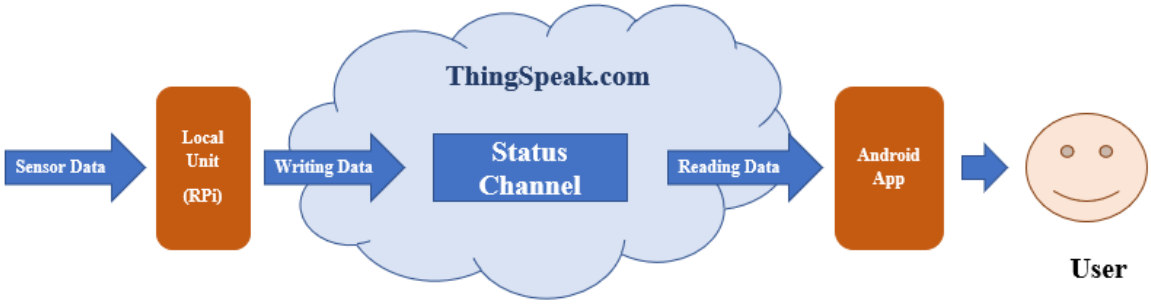


Figure 38 shows ThingSpeak as a middle layer between the local unit and the android application

We used ThingSpeak in the middle to collect data from sensors and then display the sensor readings on a custom Android app running on a mobile phone. we used the MIT App Inventor to create a custom Android app to see the sensor data and status of the system. This will help us easily access hardware to build a proof-of-concept IoT system to monitor our patient's health. [51]

MIT App Inventor

is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It is free and open-source software allows newcomers to computer programming to create application software (apps) for two operating systems (OS): Android, and iOS.

It uses a graphical user interface (GUI) very similar to the programming languages Scratch (programming language), which allows users to drag and drop visual objects to create an application that can run on Android devices. [52]

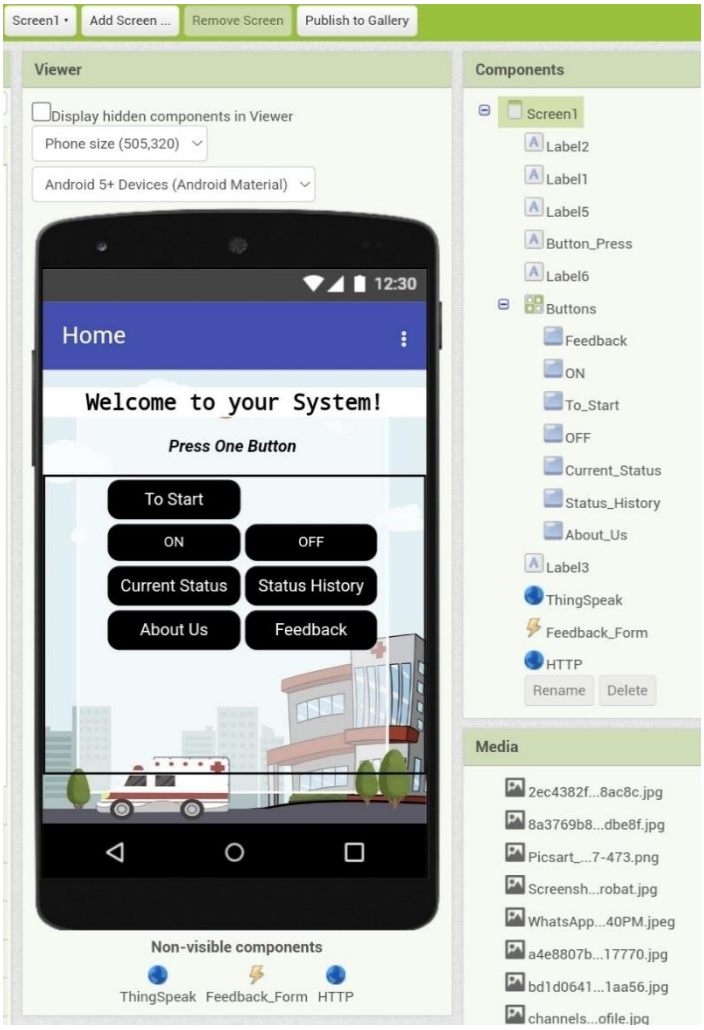
Regarding having this project on business, each device produced by the company will have its own QR code, this code will carry the application made by MIT App Inventor that will ensure the security of the patient's information as every application is customized with specific links attached to the patient's information page in ThingSpeak.

The application is used to show the user how to use the device and his vital health information. On this application we provided two view options, the current status view and the status history view, that will help the user to know his data with no need to access ThingSpeak.

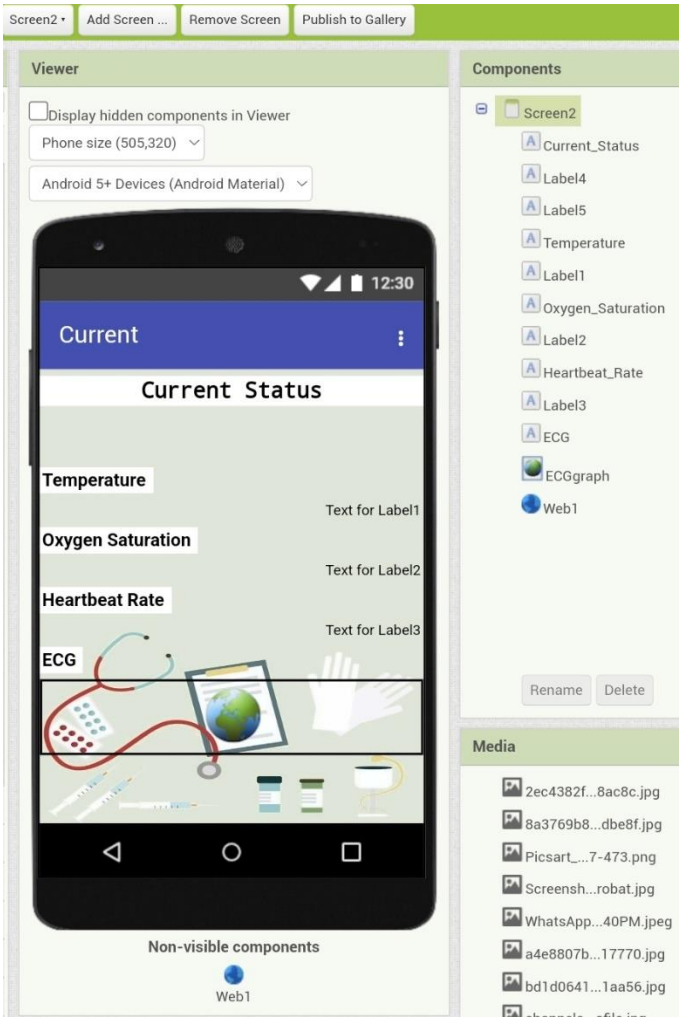
The current status view shows the values of the patient's last measure for his body temperature, oxygen saturation level, heartbeat rate and ECG graph while the status history view shows a graph for all measured values of his body temperature, oxygen saturation level and heartbeat rate since the first device usage.

In addition, we added a promotion video to help the user know the project capabilities and how to use it.

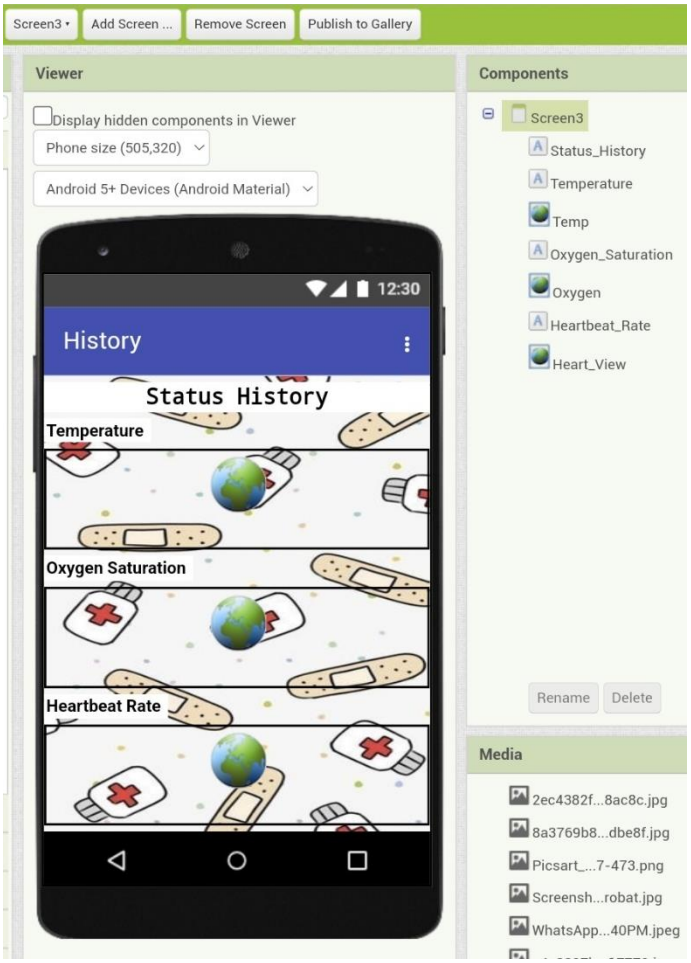
The following figures shows the application screens:



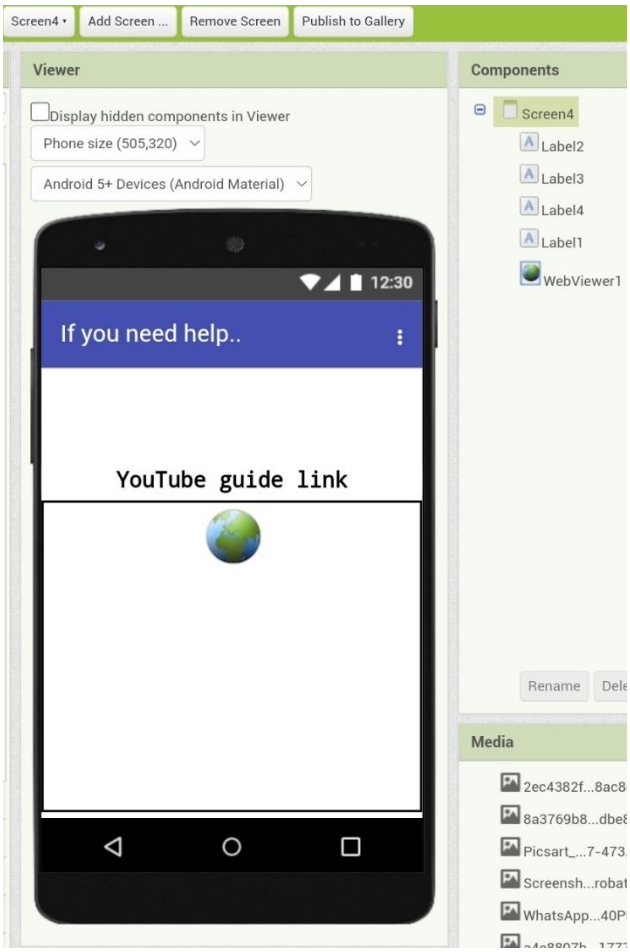
b) screen 1



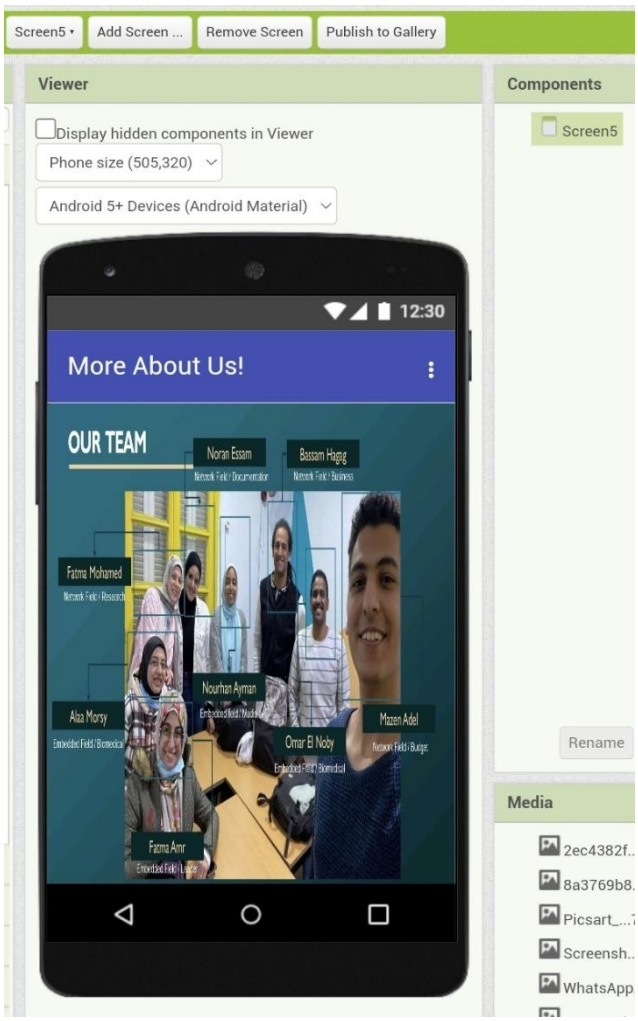
a) screen 2



d) screen 3

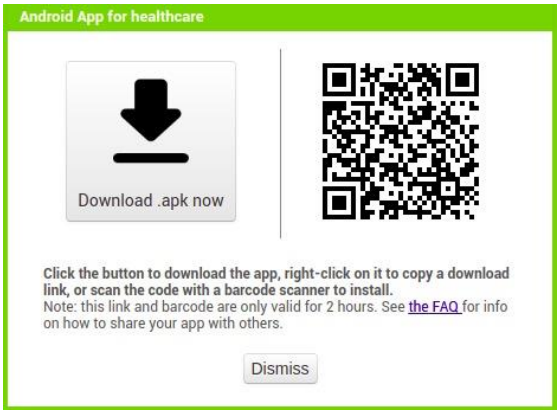


c) screen 4



e) screen 5

This application is a prototype for the actual application that can be downloaded scanning a QR code as shown in the figure below



*See the MIT APP code blocks on Appendix (D)

CHAPTER 6

Future Work

Finally, we would like to end our talking about this project by highlighting our discussions and brainstorming through all the meetings we had together as a cooperative team, during all the meetings we had we discussed many ideas about how the patient will use our device and if it will be easy for him to use it alone or he will need someone's help, also we discussed how we can facilitate the use of the device on our patient by designing it in a shape so he can wear it easily whenever he want to use it and wherever he is, so let's take a close look and explain more.

Future design of the device

In our brainstorming meetings we had before working practically on our device, we have discussed many ideas for its design and how it can look like, until we reached an idea which will facilitate the patient use while sitting on his chair, this idea was designing a ***glove,*** but how we can fit our device on this glove, this is what we are going to explain now.

The idea was placing the brain of our device which is the ***Raspberry pi*** in the middle of the glove which considers the palm of the patient's hand, where sensors will be placed on fingers of the glove or connected through wires to the body as the figure below clarifying.

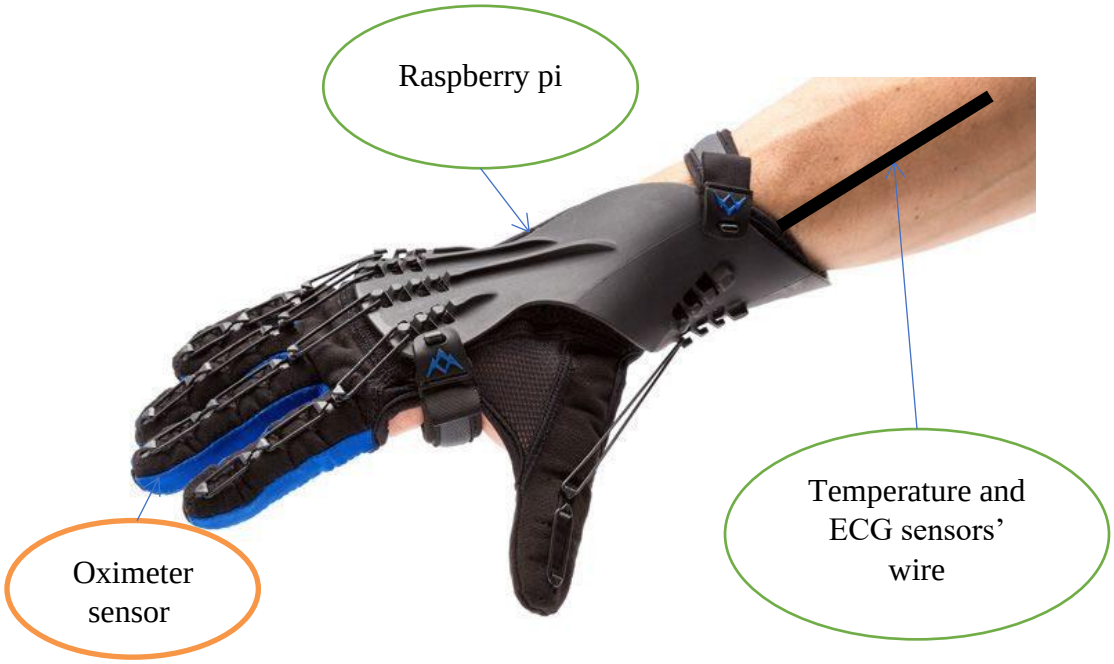


Figure 39 shows future design of the project

In our project we used only three sensors beside fall detection, but we still can add more sensors, and this considers one of our future working for our device as an upgraded version.

Due to some circumstances, we couldn't afford this design for our device for the current moment we just settle for PCBs to connect the device all together, but definitely it is going to be in our future plan for upgrading it.

Future of fall detection design

Making fall detection as an independent device is important as it allows the patient to use it daily independent of the whole project, so we came up with an idea of placing it around the patient's leg.

The fall detection's sensor, microcontroller, and connections are placed inside a box which is connected to a scotch so the patient can wrap it around his leg and remove it easily whenever he wants, but because of calibration we found that in order to get accurate readings the device should be placed around his waist or inside his pocket. So, in our future plan we aim to make use of it like the idea we mentioned above in order to introduce a new technique that is not available till now, maybe we can also upgrade this idea more and more by building it in a shoe which made especially for elderly patients. If we really managed to create something like this, it will definitely make a fuss in healthcare media.

Availability of the device

Using the design we already mentioned, will be so easy for every patient to have the device in his home, and it will facilitate on him to use it without facing any troubles.

By connecting our device with ***ThingSpeak*** cloud, each patient can have his own channel, where each channel has a specific API key, which means that each patient has a unique API key.

At first our device had only one API key which limited the number of patients that can use the device, in another word the device was made for just single patient and can't be used by any other patient, and this was considered a disadvantage in our device as for example if there are two patients living in the same home, they have to buy two devices because both of them can't use the same device and that will cost them more money which is not good at all, so we came up with two options that solved this problem, and they are:

- The first option is that when a patient uses the device, he inserts his own API key which leads him to his own channel.
- The second option is an application we created to help us solving that problem, this application is MIT app, which was already discussed briefly in previous chapter, there is a feature in this app where it creates a unique QR code for each user whenever he wants to use the device as it also leads him to his own channel.

Accessibility of patients within doctors

In our project, with the help of our cloud ***ThingSpeak*** we succeeded to create communication path between patients and doctors, which ease on patients to have a backup for all their readings where they can take a look at them whenever they like to.

On the other hand, it ease on doctors as well to take a look at all the patient's reading through a specific period of time to detect whether this patient's condition is stable or need an urgent visit to the hospital.

The doctor can access more than one channel, in another word he can have access more than one patient's readings at the same time, and that will make it easy for a single doctor to manage and communicate with more than one patient.

In our future plan regarding this topic, we are thinking of creating an application where the patient can choose any hospital, he would like to pursue his health state with, he can also choose a specific doctor who he would prefer to interact with according to ***rating principle***.

Rating principle

We are going to add rating feature for every doctor in each hospital, so when a patient interacts with a certain doctor, he can give him a rating depends on whether this doctor was good or bad, and this will help other patients who have no idea about either doctors' qualifications or their sociable skills with patients to choose according to this rating.

This will help patients also to feel reassurance toward the doctor they are interacting with, and that's what we are aiming to achieve.

References

[1] S. T. B. B. S. M. Krishna Chandra Devkota, "ECG findings in elderly," *ResearchGate*, vol. 8, no. 2, pp. 128-32, Jul 2006.

[2] Y. Z. X. C. Y. F. Yuehong YIN, "The internet of things in healthcare: An overview," *Journal of Industrial Information Integration*, vol. 1, no. 1, pp. 3-13, March 2016.

[3] A. S. Gillis, "TechTarget," 2022. [Online]. Available: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT.

[4] "CONNECTED DEVELOPMENT," 2022. [Online]. Available: https://www.connecteddev.com/healthcare.

[5] Harshith, "IOTEDU," 10 Apr 2020. [Online]. Available: https://iot4beginners.com/applications-of-internet-of-things-in-healthcare/.

[6] "Aarushi@," 26 June 2021. [Online]. Available: https://aarushiaarushi.blogspot.com/2021/06/role-of-iot-in-health-care.html.

[7] M. Gudino, "Arrow," 26 Feb 2018. [Online]. Available: https://www.arrow.com/en/research-and-events/articles/engineering-basics-what-is-a-microcontroller. [Accessed 2022].

[8] "components101," 23 October 2019. [Online]. Available: https://components101.com/articles/difference-between-microprocessor-and-microcontroller#:~:text=The%2032-bit%20microprocessor%20can%20handle%2032-bit%20binary%20data,are%208%20bit%2C%2016%20bit%20or%2032%20bit.. [Accessed 2022].

[9] S. Jena, "GeeksForGeeks," 27 Sep 2020. [Online]. Available: https://www.geeksforgeeks.org/difference-between-mcu-and-soc/.

[10] AnySilicon, "anysilicon," [Online]. Available: https://anysilicon.com/what-is-a-system-on-chip-soc/. [Accessed July 2022].

[11] G. H. Eben Upton, Raspberry Pi User Guide, 2012.

[12] "Vector Four Engineering," [Online]. Available: https://vector4engineering.com/product/pi-3-model-b/. [Accessed 2022].

[13] "components101," 26 April 2018. [Online]. Available: https://components101.com/microcontrollers/raspberry-pi-3-pinout-features-datasheet. [Accessed 2022].

[14] M. Yuan, "IBM Developer," IBM, 6 Augest 2017. [Online]. Available: https://developer.ibm.com/tutorials/iot-nodemcu-open-why-use/. [Accessed 2022].

[15] ElectroPeak, "PROJECT UUB," 14 August 2019. [Online]. Available: https://create.arduino.cc/projecthub/electropeak/getting-started-w-nodemcu-esp8266-on-arduino-ide-28184f.

[16] N. Team, "NodeMcu," 2018. [Online]. Available: https://www.nodemcu.com/index_en.html#fr_54747661d775ef1a3600009e.

[17] "IoTDESIGN PRO," 30 Septemper 2019. [Online]. Available: https://iotdesignpro.com/articles/arduino-nano-vs-raspberry-pi-zero-w-vs-nodemcu-for-iot-projects. [Accessed 2022].

[18] S. Campbell, "Circuit Basics," 2016. [Online]. Available: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/. [Accessed 2022].

[19] "vmware," vmware, 2022. [Online]. Available: https://www.vmware.com/topics/glossary/content/cloud-networking.html. [Accessed 2022].

[20] "NetworkCoverage," 2021. [Online]. Available: https://www.netcov.com/what-is-cloud-computing/. [Accessed 2022].

[21] D. Rana, "Dzone," 19 May 2022. [Online]. Available: https://dzone.com/articles/10-cloud-platforms-for-internet-of-things-iot.

[22] "ThingSpeak," MathWorks, 2022. [Online]. Available: https://thingspeak.com/pages/learn_more.

[23] E. Moreau, "lifewire," 31 August 2021. [Online]. Available: https://www.lifewire.com/what-is-ifttt-4172417. [Accessed 2022].

[24] M. K. K. A. G. W.-K. C. A. Irving H Gomolin, "Older Is Colder: Temperature Range and Variation in Older People," *Journal of the American Geriatrics Society,* vol. 53, no. 2, pp. 2170-2, January 2006.

[25] R. N. S. C. Tanya Leinicke, "Fever In The Elderly: How To Surmount The Unique Diagnostic And Therapeutic Challenges," 1 October 1999. [Online]. Available: https://www.ebmedicine.net/topics/infectious-disease/fever-elderly. [Accessed 2022].

[26] R. Ansorge, "WebMD," July 2016. [Online]. Available: https://www.webmd.com/first-aid/fevers-causes-symptoms-treatments#:~:text=The%20most%20common%20causes%20of,Heat%20exhaustion. [Accessed 2022].

[27] A. Anglia, Using a pulse oximeter To check you are OK, NHS England, 2022.

[28] A. Gotter, "healthline," 5 August 2021. [Online]. Available: https://www.healthline.com/health/pulse-oximetry#how-it-works. [Accessed 2022].

[29] "maxim integrated," Analog Devices, 2022. [Online]. Available: https://www.maximintegrated.com/en/products/interface/sensor-interface/MAX30105.html.

[30] E. R. R. S. S Bakhri, "Design of Low Cost Pulse Oximetry Based on Raspberry Pi," *Journal of Physics Conference Series,* 2020.

[31] Aktualisiert, "cora health," 1 July 2019. [Online]. Available: https://www.cora.health/guide/resting-heart-rate/#:~:text=The%20pulse%20is%20very%20much%20dependent. [Accessed 2022].

[32] MolecularD, "instructables circuits," [Online]. Available: https://www.instructables.com/Pulse-Oximeter-With-Much-Improved-Precision/. [Accessed July 2022].

[33] "Healthily," NHS England, August 2022. [Online]. Available: https://www.livehealthily.com/heart-brain-monitoring/electrocardiogram.

[34] "National Library of Medicine," Institute for Quality and Efficiency in Health Care, 31 January 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK536878/. [Accessed 2022].

[35] A. S. D. M. K. Gadó, "Most common cardiovascular diseases of the elderly – A review article," *AKJournals,* vol. 4, no. 2, p. 27–32, 2022.

[36] "components101," 13 July 2020. [Online]. Available: https://components101.com/modules/ad8232-ecg-module. [Accessed 2022].

[37] "How To Electronics," 26 November 2020. [Online]. Available: https://how2electronics.com/ecg-monitoring-with-ad8232-ecg-sensor-arduino/. [Accessed 2022].

[38] C. Staff, "circuitschools," 18 February 2022. [Online]. Available: https://www.circuitschools.com/ecg-monitoring-system-using-ad8232-with-arduino-or-esp32-iot-based/#:~:text=The%20AD8232%20module%20allows%20recording%20the%20electrical%20activity,specific%20pathways%20within%20the%20heart%2C%20causing%20the%20heartbeat..

[39] T. DiCola, "adafruit," 9 February 2016. [Online]. Available: https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/ads1015-slash-ads1115. [Accessed 2022].

[40] M. Gudino, "Arrow," 17 April 2018. [Online]. Available: https://www.arrow.com/en/research-and-events/articles/engineering-resource-basics-of-analog-to-digital-converters#:~:text=ADCs%20follow%20a%20sequence%20when%20converting%20analog%20signals,the%20ADC%20are%20its%20sampling%20rate%20and%20resolution..

[41] "components101," 21 September 2021. [Online]. Available: https://components101.com/ics/ads1115-analog-to-digital-converter-ic-pinout-datasheet-equivalent-circuit-specs. [Accessed 2022].

[42] L. N. W. W. K. K. Arkham Zahri Rakhman, "Fall Detection System Using Accelerometer and Gyroscope Based on Smartphone," *International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE),* November 2014.

[43] H. Z. Y. Z. H. Z. Falin Wu, "Development of a Wearable-Sensor-Based Fall Detection System," *International Journal of Telemedicine and Applications,* 2015.

[44] R. Watson, "Maker PRO," 17 September 2019. [Online]. Available: https://maker.pro/raspberry-pi/tutorial/how-to-interface-an-imu-sensor-with-a-raspberry-pi. [Accessed 2022].

[45] mjwhite8119, "Programming Robots," 26 July 2019. [Online]. Available: https://mjwhite8119.github.io/Robots/mpu6050. [Accessed July 2022].

[46] "Microcontrollerslab," 2022. [Online]. Available: https://microcontrollerslab.com/esp32-mpu9250-tutorial/. [Accessed 2022].

[47] "components101," 10 February 2021. [Online]. Available: https://components101.com/sensors/MPU9250-9-dof-mems-sensor-module-datasheet-pinout-features-working. [Accessed 2022].

[48] K. ANIPINDI, "CODE Project," 23 November 2014. [Online]. Available: https://www.codeproject.com/articles/845538/an-introduction-to-thingspeak. [Accessed 2022].

[49] d. industries, "Instructables Circuits," Autodesk, 2022. [Online]. Available: https://www.instructables.com/ThingSpeak-Temperature-Monitor-with-Raspberry-Pi/.

[50] "Microcontrollerslab," 2022. [Online]. Available: https://microcontrollerslab.com/esp32-fall-detection-mpu6050-email-alerts/.

[51] "IOTDESIGN PRO," 6 July 2020. [Online]. Available: https://iotdesignpro.com/projects/iot-based-fall-detection-system-using-nodemcu-esp8266-and-accelerometer-mpu6050.

[52] H. Scharler, "MathWorks," 20 December 2017. [Online]. Available: https://blogs.mathworks.com/iot/2017/12/20/learn-how-to-build-a-custom-android-app-for-a-thingspeak-iot-project/.

[53] L. Hardesty, "Massachusetts Institute of Technology," 19 August 2010. [Online]. Available: https://news.mit.edu/2010/android-abelson-0819. [Accessed 2022].

[55] "Tutorialspoint," 2022. [Online]. Available: https://www.tutorialspoint.com/microprocessor/microcontrollers_overview.htm.

[56] "Engineering360," globalspec, [Online]. Available: https://www.globalspec.com/learnmore/semiconductors/programmable_logic/system_on_a_chip#:~:text=System%20on%20a%20chip%20%28SoC%29%20devices%20are%20semiconductor,phones%20for%20radio%20frequency%20%28RF%29%20and%20wireless%20communications.. [Accessed 2022].

Appendix A

MODEL	PROCESSOR	RAM	CONNECTIVITY	USB	HDMI	POWER
Zero	BCM2835	512MB	None	Micro USB OTG	Mini HDMI	Micro USB
Zero W	BCM2835	512MB	802.11 b/g/n wireless LAN	Micro USB OTG	Mini HDMI	Micro USB
Raspberry Pi 1 Model A+	BCM2835	512MB	None	1x USB 2.0	Full-size HDMI	Micro USB
Raspberry Pi 1 Model B+	BCM2835	512MB	100 base ethernet	4x USB 2.0	Full-size HDMI	Micro USB
Raspberry Pi 3 Model A+	BCM2837B0	512MB	dual-band wireless, Bluetooth 4.2	1x USB 2.0	Full-size HDMI	5V DC via Micro USB
Raspberry Pi 3 Model B	BCM2837	1GB	ethernet, wireless, BLE	4x USB 2.0	Full-size HDMI	2.1A via Micro USB
Raspberry Pi 3 Model B+	BCM2837B0	1GB	dual-band wireless, Bluetooth 4.2, BLE	4x USB 2.0	Full-size HDMI	5V DC via Micro USB & Power-over-Ethernet (PoE)
Raspberry Pi 4 Model B	BCM2711	2GB, 4GB, or 8GB	Gigabit ethernet, dual-band wireless, Bluetooth	2x USB 3.0 & 2x USB 2.0	2x micro-HDMI	5V DC via USB-C
Raspberry Pi 400	BCM2711	4GB	Gigabit ethernet, dual-band wireless, Bluetooth	2x USB 3.0 & 1x USB 2.0	2x micro-HDMI	5V DC via USB

Appendix B

File (1): main.py

```
from heartrate_monitor import HeartRateMonitor
import time
import argparse
from ECG_Readings import ECg_Readings

parser = argparse.ArgumentParser(description="Read and print data from MAX30102")
parser.add_argument("-r", "--raw", action="store_true",
                    help="print raw data instead of calculation result")
parser.add_argument("-t", "--time", type=int, default=30,
                    help="duration in seconds to read from sensor, default 30")
args = parser.parse_args()

while True:
    print('sensors starting...')
    hrm = HeartRateMonitor(print_raw=args.raw, print_result=(not args.raw))
    hrm.start_sensor()
    time.sleep(15)
    ECG = ECg_Readings()
    try:
        time.sleep(args.time)
    except KeyboardInterrupt:
        print('keyboard interrupt detected, exiting...')

    hrm.stop_sensor()
    #print('sensor stoped!')

```

File (2): heartrate_monitor.py

```
from max30102 import MAX30102
from ECG_Readings import ECg_Readings
from w1thermsensor import W1ThermSensor
from time import sleep
import RPi.GPIO as GPIO
import hrcalc
import threading
import time
import numpy as np
import urllib3
import requests
import sys
import datetime


baseURL = 'http://api.thingspeak.com/update?api_key=EGGDD5XE4A35W6MB'

class HeartRateMonitor(object):
    """
    A class that encapsulates the max30102 device into a thread
    """

    LOOP_TIME = 0.5

    def __init__(self, print_raw=False, print_result=False):
        self.bpm = 0
        if print_raw is True:
            print('IR, Red')
        self.print_raw = print_raw
        self.print_result = print_result

    def run_sensor(self):
        temp_sensor = W1ThermSensor()                 # create an object to store a
connection the sensor.
        temperature = temp_sensor.get_temperature()   # get the current temperature
from the DS18B20 sensor,
        sensor = MAX30102()
        ir_data = []
        red_data = []
        bpms = []
        # run until told to stop
        while not self._thread.stopped:
```

```
            # check if any data is available
            num_bytes = sensor.get_data_present()
            if num_bytes > 0:
                # grab all the data and stash it into arrays
                while num_bytes > 0:
                    red, ir = sensor.read_fifo()
                    num_bytes -= 1
                    ir_data.append(ir)
                    red_data.append(red)
                    if self.print_raw:
                        print("{0}, {1}".format(ir, red))

                while len(ir_data) > 100:
                    ir_data.pop(0)
                    red_data.pop(0)

                if len(ir_data) == 100:
                    bpm, valid_bpm, spo2, valid_spo2 =
hrcalc.calc_hr_and_spo2(ir_data, red_data)
                    if valid_bpm:
                        bpms.append(bpm)
                        while len(bpms) > 4:
                            bpms.pop(0)
                        self.bpm = np.mean(bpms)
                        if (np.mean(ir_data) < 50000 and np.mean(red_data) < 50000):
                            self.bpm = 0
                            if self.print_result:
                                print("Finger not detected")
                        if self.print_result:

                            if temperature > 38 or temperature < 35 or spo2 < 92 or
self.bpm < 60 or self.bpm > 100:                    # temperature threshold
                                # *** take an action regarding the cloud ***
                                report = {}
                                report["value1"] = temperature
                                report["value2"] = self.bpm
                                report["value3"] = spo2
                                requests.post('https://maker.ifttt.com/trigger/123/w
ith/key/nRm3Yf4BU-Irdwh4MnqoXmsH4SoHxY7wf7EpsgvFV26',data=report)
                            print("BPM: {0}, SpO2: {1}".format(self.bpm, spo2))
                            print("The temperature is %s celsius" %temperature)

                            if temperature > 34:
                                if spo2 == -999:
                                    print("Error in MAX30102 sensor")
```

```
                                    finalurl = baseURL +"&field4=%s" %(temperature)
                                    http = urllib3.PoolManager()
                                    f = http.request('GET',finalurl)
                                    f.read()
                                    f.close()
                                else:
                                    finalurl = baseURL
+"&field2=%s&field3=%s&field4=%s" %(self.bpm,spo2,temperature)
                                    http = urllib3.PoolManager()
                                    f = http.request('GET',finalurl)
                                    f.read()
                                    f.close()
                            else:
                                 if spo2 == -999:
                                    print("Nothing detected try again")
                                 else:
                                    print("Temperature sensor not detect and body")
                                    finalurl = baseURL +"&field2=%s&field3=%s"
%(self.bpm,spo2)
                                    http = urllib3.PoolManager()
                                    f = http.request('GET',finalurl)
                                    f.read()
                                    f.close()


            #time.sleep(self.LOOP_TIME)
            #time.sleep(0.001)

        #sensor.shutdown()

    def start_sensor(self):
        self._thread = threading.Thread(target=self.run_sensor)
        self._thread.stopped = False
        self._thread.start()

    def stop_sensor(self, timeout=2.0):
        self._thread.stopped = True
        self.bpm = 0
        self._thread.join(timeout)
```

File (3): max30102.py

```
# -*-coding:utf-8-*-

# this code is currently for python 2.7
from __future__ import print_function
from time import sleep
import smbus

# register addresses
REG_INTR_STATUS_1 = 0x00
REG_INTR_STATUS_2 = 0x01

REG_INTR_ENABLE_1 = 0x02
REG_INTR_ENABLE_2 = 0x03

REG_FIFO_WR_PTR = 0x04
REG_OVF_COUNTER = 0x05
REG_FIFO_RD_PTR = 0x06
REG_FIFO_DATA = 0x07
REG_FIFO_CONFIG = 0x08

REG_MODE_CONFIG = 0x09
REG_SPO2_CONFIG = 0x0A
REG_LED1_PA = 0x0C

REG_LED2_PA = 0x0D
REG_PILOT_PA = 0x10
REG_MULTI_LED_CTRL1 = 0x11
REG_MULTI_LED_CTRL2 = 0x12

REG_TEMP_INTR = 0x1F
REG_TEMP_FRAC = 0x20
REG_TEMP_CONFIG = 0x21
REG_PROX_INT_THRESH = 0x30
REG_REV_ID = 0xFE
REG_PART_ID = 0xFF


class MAX30102():
    # by default, this assumes that the device is at 0x57 on channel 1
    def __init__(self, channel=1, address=0x57):
        #print("Channel: {0}, address: {1}".format(channel, address))
        self.address = address
```

```
41          self.channel = channel
42          self.bus = smbus.SMBus(self.channel)
43
44          self.reset()
45
46          sleep(1)  # wait 1 sec
47
48          # read & clear interrupt register (read 1 byte)
49          reg_data = self.bus.read_i2c_block_data(self.address, REG_INTR_STATUS_1, 1)
50          # print("[SETUP] reset complete with interrupt register0:
51  {0}".format(reg_data))
52          self.setup()
53          # print("[SETUP] setup complete")
54
55      def shutdown(self):
56          """
57          Shutdown the device.
58          """
59          self.bus.write_i2c_block_data(self.address, REG_MODE_CONFIG, [0x80])
60
61      def reset(self):
62          """
63          Reset the device, this will clear all settings,
64          so after running this, run setup() again.
65          """
66          self.bus.write_i2c_block_data(self.address, REG_MODE_CONFIG, [0x40])
67
68      def setup(self, led_mode=0x03):
69          """
70          This will setup the device with the values written in sample Arduino code.
71          """
72          # INTR setting
73          # 0xc0 : A_FULL_EN and PPG_RDY_EN = Interrupt will be triggered when
74          # fifo almost full & new fifo data ready
75          self.bus.write_i2c_block_data(self.address, REG_INTR_ENABLE_1, [0xc0])
76          self.bus.write_i2c_block_data(self.address, REG_INTR_ENABLE_2, [0x00])
77
78          # FIFO_WR_PTR[4:0]
79          self.bus.write_i2c_block_data(self.address, REG_FIFO_WR_PTR, [0x00])
80          # OVF_COUNTER[4:0]
81          self.bus.write_i2c_block_data(self.address, REG_OVF_COUNTER, [0x00])
82          # FIFO_RD_PTR[4:0]
83          self.bus.write_i2c_block_data(self.address, REG_FIFO_RD_PTR, [0x00])
84
85          # 0b 0100 1111
```

```
86          # sample avg = 4, fifo rollover = false, fifo almost full = 17
87          self.bus.write_i2c_block_data(self.address, REG_FIFO_CONFIG, [0x4f])
88
89          # 0x02 for read-only, 0x03 for SpO2 mode, 0x07 multimode LED
90          self.bus.write_i2c_block_data(self.address, REG_MODE_CONFIG, [led_mode])
91          # 0b 0010 0111
92          # SPO2_ADC range = 4096nA, SPO2 sample rate = 100Hz, LED pulse-width = 411uS
93          self.bus.write_i2c_block_data(self.address, REG_SPO2_CONFIG, [0x27])
94
95          # choose value for ~7mA for LED1
96          self.bus.write_i2c_block_data(self.address, REG_LED1_PA, [0x24])
97          # choose value for ~7mA for LED2
98          self.bus.write_i2c_block_data(self.address, REG_LED2_PA, [0x24])
99          # choose value fro ~25mA for Pilot LED
100         self.bus.write_i2c_block_data(self.address, REG_PILOT_PA, [0x7f])
101
102     # this won't validate the arguments!
103     # use when changing the values from default
104     def set_config(self, reg, value):
105         self.bus.write_i2c_block_data(self.address, reg, value)
106
107     def get_data_present(self):
108         read_ptr = self.bus.read_byte_data(self.address, REG_FIFO_RD_PTR)
109         write_ptr = self.bus.read_byte_data(self.address, REG_FIFO_WR_PTR)
110         if read_ptr == write_ptr:
111             return 0
112         else:
113             num_samples = write_ptr - read_ptr
114             # account for pointer wrap around
115             if num_samples < 0:
116                 num_samples += 32
117             return num_samples
118
119     def read_fifo(self):
120         """
121         This function will read the data register.
122         """
123         red_led = None
124         ir_led = None
125
126         # read 1 byte from registers (values are discarded)
127         reg_INTR1 = self.bus.read_i2c_block_data(self.address, REG_INTR_STATUS_1, 1)
128         reg_INTR2 = self.bus.read_i2c_block_data(self.address, REG_INTR_STATUS_2, 1)
129
130         # read 6-byte data from the device
```

```
131         d = self.bus.read_i2c_block_data(self.address, REG_FIFO_DATA, 6)
132 
133         # mask MSB [23:18]
134         red_led = (d[0] << 16 | d[1] << 8 | d[2]) & 0x03FFFF
135         ir_led = (d[3] << 16 | d[4] << 8 | d[5]) & 0x03FFFF
136 
137         return red_led, ir_led
138 
139     def read_sequential(self, amount=100):
140         """
141         This function will read the red-led and ir-led `amount` times.
142         This works as blocking function.
143         """
144         red_buf = []
145         ir_buf = []
146         count = amount
147         while count > 0:
148             num_bytes = self.get_data_present()
149             while num_bytes > 0:
150                 red, ir = self.read_fifo()
151 
152                 red_buf.append(red)
153                 ir_buf.append(ir)
154                 num_bytes -= 1
155                 count -= 1
156 
157         return red_buf, ir_buf
158 
```

File (4): hrcalc.py

```
# -*-coding:utf-8

import numpy as np
from time import sleep
# 25 samples per second (in algorithm.h)


SAMPLE_FREQ = 25
# taking moving average of 4 samples when calculating HR
# in algorithm.h, "DONOT CHANGE" comment is attached
MA_SIZE = 4
# sampling frequency * 4 (in algorithm.h)
BUFFER_SIZE = 100


# this assumes ir_data and red_data as np.array
def calc_hr_and_spo2(ir_data, red_data):
    """
    By detecting  peaks of PPG cycle and corresponding AC/DC
    of red/infra-red signal, the an_ratio for the SPO2 is computed.
    """
    # get dc mean
    ir_mean = int(np.mean(ir_data))

    # remove DC mean and inver signal
    # this lets peak detecter detect valley
    x = -1 * (np.array(ir_data) - ir_mean)

    # 4 point moving average
    # x is np.array with int values, so automatically casted to int
    for i in range(x.shape[0] - MA_SIZE):
        x[i] = np.sum(x[i:i+MA_SIZE]) / MA_SIZE

    # calculate threshold
    n_th = int(np.mean(x))
    n_th = 30 if n_th < 30 else n_th  # min allowed
    n_th = 60 if n_th > 60 else n_th  # max allowed

    ir_valley_locs, n_peaks = find_peaks(x, BUFFER_SIZE, n_th, 4, 15)
    # print(ir_valley_locs[:n_peaks], ",", end="")
    peak_interval_sum = 0
    if n_peaks >= 2:
```

```
        for i in range(1, n_peaks):
            peak_interval_sum += (ir_valley_locs[i] - ir_valley_locs[i-1])
        peak_interval_sum = int(peak_interval_sum / (n_peaks - 1))
        hr = int(SAMPLE_FREQ * 60 / peak_interval_sum)
        hr_valid = True

        sleep(10)
    else:
        hr = -999  # unable to calculate because # of peaks are too small
        hr_valid = False


    # ---------spo2---------

    # find precise min near ir_valley_locs (???)
    exact_ir_valley_locs_count = n_peaks

    # find ir-red DC and ir-red AC for SPO2 calibration ratio
    # find AC/DC maximum of raw

    # FIXME: needed??
    for i in range(exact_ir_valley_locs_count):
        if ir_valley_locs[i] > BUFFER_SIZE:
            spo2 = -999  # do not use SPO2 since valley loc is out of range
            spo2_valid = False
            return hr, hr_valid, spo2, spo2_valid

    i_ratio_count = 0
    ratio = []

    # find max between two valley locations
    # and use ratio between AC component of Ir and Red DC component of Ir and Red
for SpO2
    red_dc_max_index = -1
    ir_dc_max_index = -1
    for k in range(exact_ir_valley_locs_count-1):
        red_dc_max = -16777216
        ir_dc_max = -16777216
        if ir_valley_locs[k+1] - ir_valley_locs[k] > 3:
            for i in range(ir_valley_locs[k], ir_valley_locs[k+1]):
                if ir_data[i] > ir_dc_max:
                    ir_dc_max = ir_data[i]
                    ir_dc_max_index = i
                if red_data[i] > red_dc_max:
                    red_dc_max = red_data[i]
```

```
                    red_dc_max_index = i

            red_ac = int((red_data[ir_valley_locs[k+1]] -
red_data[ir_valley_locs[k]]) * (red_dc_max_index - ir_valley_locs[k]))
            red_ac = red_data[ir_valley_locs[k]] + int(red_ac / (ir_valley_locs[k+1]
- ir_valley_locs[k]))
            red_ac = red_data[red_dc_max_index] - red_ac  # subtract linear DC
components from raw

            ir_ac = int((ir_data[ir_valley_locs[k+1]] - ir_data[ir_valley_locs[k]])
* (ir_dc_max_index - ir_valley_locs[k]))
            ir_ac = ir_data[ir_valley_locs[k]] + int(ir_ac / (ir_valley_locs[k+1] -
ir_valley_locs[k]))
            ir_ac = ir_data[ir_dc_max_index] - ir_ac  # subtract linear DC
components from raw

            nume = red_ac * ir_dc_max
            denom = ir_ac * red_dc_max
            if (denom > 0 and i_ratio_count < 5) and nume != 0:
                # original cpp implementation uses overflow intentionally.
                # but at 64-bit OS, Pyhthon 3.X uses 64-bit int and nume*100/denom
does not trigger overflow
                # so using bit operation ( &0xffffffff ) is needed
                ratio.append(int(((nume * 100) & 0xffffffff) / denom))
                i_ratio_count += 1

    # choose median value since PPG signal may vary from beat to beat
    ratio = sorted(ratio)  # sort to ascending order
    mid_index = int(i_ratio_count / 2)

    ratio_ave = 0
    if mid_index > 1:
        ratio_ave = int((ratio[mid_index-1] + ratio[mid_index])/2)
    else:
        if len(ratio) != 0:
            ratio_ave = ratio[mid_index]

    # why 184?
    # print("ratio average: ", ratio_ave)
    if ratio_ave > 2 and ratio_ave < 184:
        # -45.060 * ratioAverage * ratioAverage / 10000 + 30.354 * ratioAverage /
100 + 94.845
        spo2 = -45.060 * (ratio_ave**2) / 10000.0 + 30.054 * ratio_ave / 100.0 +
94.845
        spo2_valid = True
```

```
131
132         else:
133             spo2 = -999
134             spo2_valid = False
135
136         return hr, hr_valid, spo2, spo2_valid
137
138     def find_peaks(x, size, min_height, min_dist, max_num):
139         """
140         Find at most MAX_NUM peaks above MIN_HEIGHT separated by at least MIN_DISTANCE
141         """
142         ir_valley_locs, n_peaks = find_peaks_above_min_height(x, size, min_height,
143     max_num)
144         ir_valley_locs, n_peaks = remove_close_peaks(n_peaks, ir_valley_locs, x,
145     min_dist)
146
147         n_peaks = min([n_peaks, max_num])
148
149         return ir_valley_locs, n_peaks
150
151     def find_peaks_above_min_height(x, size, min_height, max_num):
152         """
153         Find all peaks above MIN_HEIGHT
154         """
155
156         i = 0
157         n_peaks = 0
158         ir_valley_locs = []  # [0 for i in range(max_num)]
159         while i < size - 1:
160             if x[i] > min_height and x[i] > x[i-1]:  # find the left edge of potential
161     peaks
162                 n_width = 1
163                 # original condition i+n_width < size may cause IndexError
164                 # so I changed the condition to i+n_width < size - 1
165                 while i + n_width < size - 1 and x[i] == x[i+n_width]:  # find flat
166     peaks
167                     n_width += 1
168                 if x[i] > x[i+n_width] and n_peaks < max_num:  # find the right edge of
169     peaks
170                     # ir_valley_locs[n_peaks] = i
171                     ir_valley_locs.append(i)
172                     n_peaks += 1  # original uses post increment
173                     i += n_width + 1
```

```
            else:
                i += n_width
        else:
            i += 1

    return ir_valley_locs, n_peaks


def remove_close_peaks(n_peaks, ir_valley_locs, x, min_dist):
    """
    Remove peaks separated by less than MIN_DISTANCE
    """

    # should be equal to maxim_sort_indices_descend
    # order peaks from large to small
    # should ignore index:0
    sorted_indices = sorted(ir_valley_locs, key=lambda i: x[i])
    sorted_indices.reverse()

    # this "for" loop expression does not check finish condition
    # for i in range(-1, n_peaks):
    i = -1
    while i < n_peaks:
        old_n_peaks = n_peaks
        n_peaks = i + 1
        # this "for" loop expression does not check finish condition
        # for j in (i + 1, old_n_peaks):
        j = i + 1
        while j < old_n_peaks:
            n_dist = (sorted_indices[j] - sorted_indices[i]) if i != -1 else
(sorted_indices[j] + 1)  # lag-zero peak of autocorr is at index -1
            if n_dist > min_dist or n_dist < -1 * min_dist:
                sorted_indices[n_peaks] = sorted_indices[j]
                n_peaks += 1  # original uses post increment
            j += 1
        i += 1

    sorted_indices[:n_peaks] = sorted(sorted_indices[:n_peaks])

    return sorted_indices, n_peaks
```

File (5): ECG_Readings.py

```
import time
import os
import array
import pylab as pl
import Adafruit_ADS1x15
import requests

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def ECg_Readings():

    print('ECG sensors starting...')
    ECG_volts=[]

    def ReadChannel(channel):
        adc = Adafruit_ADS1x15.ADS1115()
        GAIN = 1
        data = adc.read_adc(0, gain=GAIN)
        return data

    # Function to convert data to voltage level,
    # rounded to specified number of decimal places.
    def ConvertVolts(data,places):
        volts = (data * 4.096) / float(32768)
        volts = round(volts,places)
        return volts
    # Define delay between readings
    for i in range (0,100):

        # Read the light sensor data
        ECG_level = ReadChannel(0)
        ECG_volts.append(ConvertVolts(ECG_level,2))
        ECG_report = {}
        ECG_report["value1"] = ECG_level
        requests.post('https://maker.ifttt.com/trigger/csv/with/key/nRm3Yf4BU-
Irdwh4MnqoXmsH4SoHxY7wf7EpsgvFV26',data=ECG_report)
        time.sleep(0.003)
    #pl.plot(ECG_volts,label="ECG")
    #pl.legend()
    #pl.show()
    return ECG_volts
```

File (6): fall_detection_with_ifttt.ino

```
#include <Wire.h>
#include <ESP8266WiFi.h>
const int MPU_addr=0x68;  // I2C address of the MPU-9250
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
float ax=0, ay=0, az=0, gx=0, gy=0, gz=0;
boolean fall = false; //stores if a fall has occurred
boolean trigger1=false; //stores if first trigger (lower threshold) has occurred
boolean trigger2=false; //stores if second trigger (upper threshold) has occurred
boolean trigger3=false; //stores if third trigger (orientation change) has occurred
byte trigger1count=0; //stores the counts past since trigger 1 was set true
byte trigger2count=0; //stores the counts past since trigger 2 was set true
byte trigger3count=0; //stores the counts past since trigger 3 was set true
int angleChange=0;
// WiFi network info.
const char *ssid =  "secret";     // Enter your WiFi Name
const char *pass =  "%$#@Basal2021@#$%"; // Enter your WiFi Password
void send_event(const char *event);
const char *host = "maker.ifttt.com";
const char *privateKey = "nRm3Yf4BU-Irdwh4MnqoXmsH4SoHxY7wf7EpsgvFV26";
void setup(){
 Serial.begin(115200);
 Wire.begin();
 Wire.beginTransmission(MPU_addr);
 Wire.write(0x6B);  // PWR_MGMT_1 register
 Wire.write(0);     // set to zero (wakes up the MPU-9250)
 Wire.endTransmission(true);
 Serial.println("Wrote to IMU");
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED)
  {
```

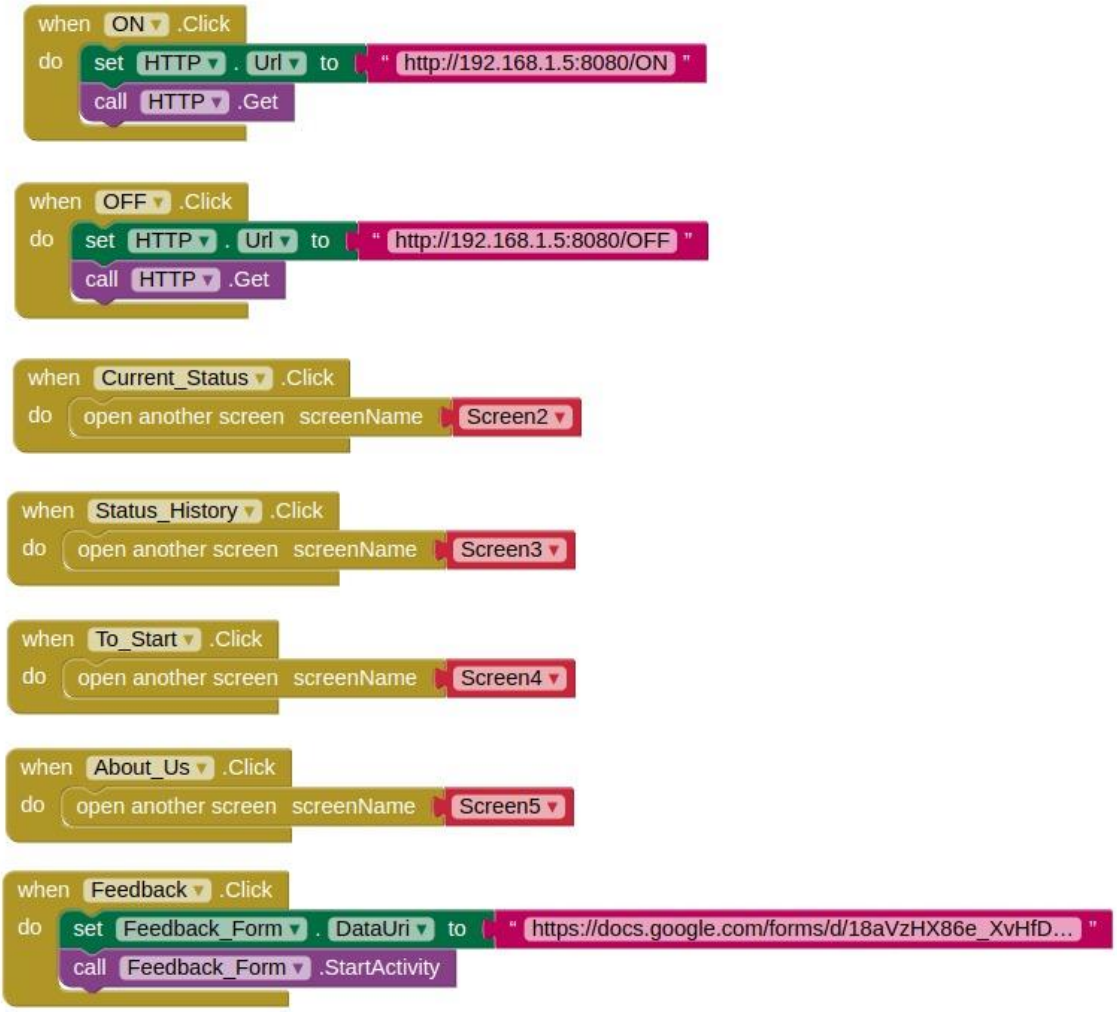
```
    delay(500);
    Serial.print(".");              // print ... till not connected
   }
   Serial.println("");
   Serial.println("WiFi connected");
  }
void loop(){
 mpu_read();
 ax = (AcX-2050)/16384.00;
 ay = (AcY-77)/16384.00;
 az = (AcZ-1947)/16384.00;
 gx = (GyX+270)/131.07;
 gy = (GyY-351)/131.07;
 gz = (GyZ+136)/131.07;
 // calculating Amplitute vactor for 3 axis
 float Raw_Amp = pow(pow(ax,2)+pow(ay,2)+pow(az,2),0.5);
 int Amp = Raw_Amp * 10;  // Mulitiplied by 10 bcz values are between 0 to 1
 Serial.println(Amp);
 if (Amp<=2 && trigger2==false){ //if AM breaks lower threshold (0.4g)
   trigger1=true;
   Serial.println("TRIGGER 1 ACTIVATED");
   }
 if (trigger1==true){
   trigger1count++;
   if (Amp>=12){ //if AM breaks upper threshold (3g)
     trigger2=true;
     Serial.println("TRIGGER 2 ACTIVATED");
     trigger1=false; trigger1count=0;
     }
 }
 if (trigger2==true){
   trigger2count++;
   angleChange = pow(pow(gx,2)+pow(gy,2)+pow(gz,2),0.5);
Serial.println(angleChange);
   if (angleChange>=30 && angleChange<=400){ //if orientation changes by between 80-
100 degrees
    trigger3=true; trigger2=false; trigger2count=0;
    Serial.println(angleChange);
    Serial.println("TRIGGER 3 ACTIVATED");
      }
   }
 if (trigger3==true){
    trigger3count++;
    if (trigger3count>=10){
       angleChange = pow(pow(gx,2)+pow(gy,2)+pow(gz,2),0.5);
```

```
        //delay(10);
        Serial.println(angleChange);
        if ((angleChange>=0) && (angleChange<=10)){ //if orientation changes remains
between 0-10 degrees
            fall=true; trigger3=false; trigger3count=0;
            Serial.println(angleChange);
              }
        else{ //user regained normal orientation
           trigger3=false; trigger3count=0;
           Serial.println("TRIGGER 3 DEACTIVATED");
        }
      }
   }
  if (fall==true){ //in event of a fall detection
    Serial.println("FALL DETECTED");
    send_event("fall_detect");
    fall=false;
    }
  if (trigger2count>=6){ //allow 0.5s for orientation change
    trigger2=false; trigger2count=0;
    Serial.println("TRIGGER 2 DECACTIVATED");
    }
  if (trigger1count>=6){ //allow 0.5s for AM to break upper threshold
    trigger1=false; trigger1count=0;
    Serial.println("TRIGGER 1 DECACTIVATED");
    }
   delay(100);
    }
void mpu_read(){
 Wire.beginTransmission(MPU_addr);
 Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
 Wire.endTransmission(false);
 Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
 AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
 AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
 AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
 Tmp=Wire.read()<<8|Wire.read();  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
 GyX=Wire.read()<<8|Wire.read();  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
 GyY=Wire.read()<<8|Wire.read();  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
 GyZ=Wire.read()<<8|Wire.read();  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
 }
void send_event(const char *event)
{
  Serial.print("Connecting to ");
  Serial.println(host);
```

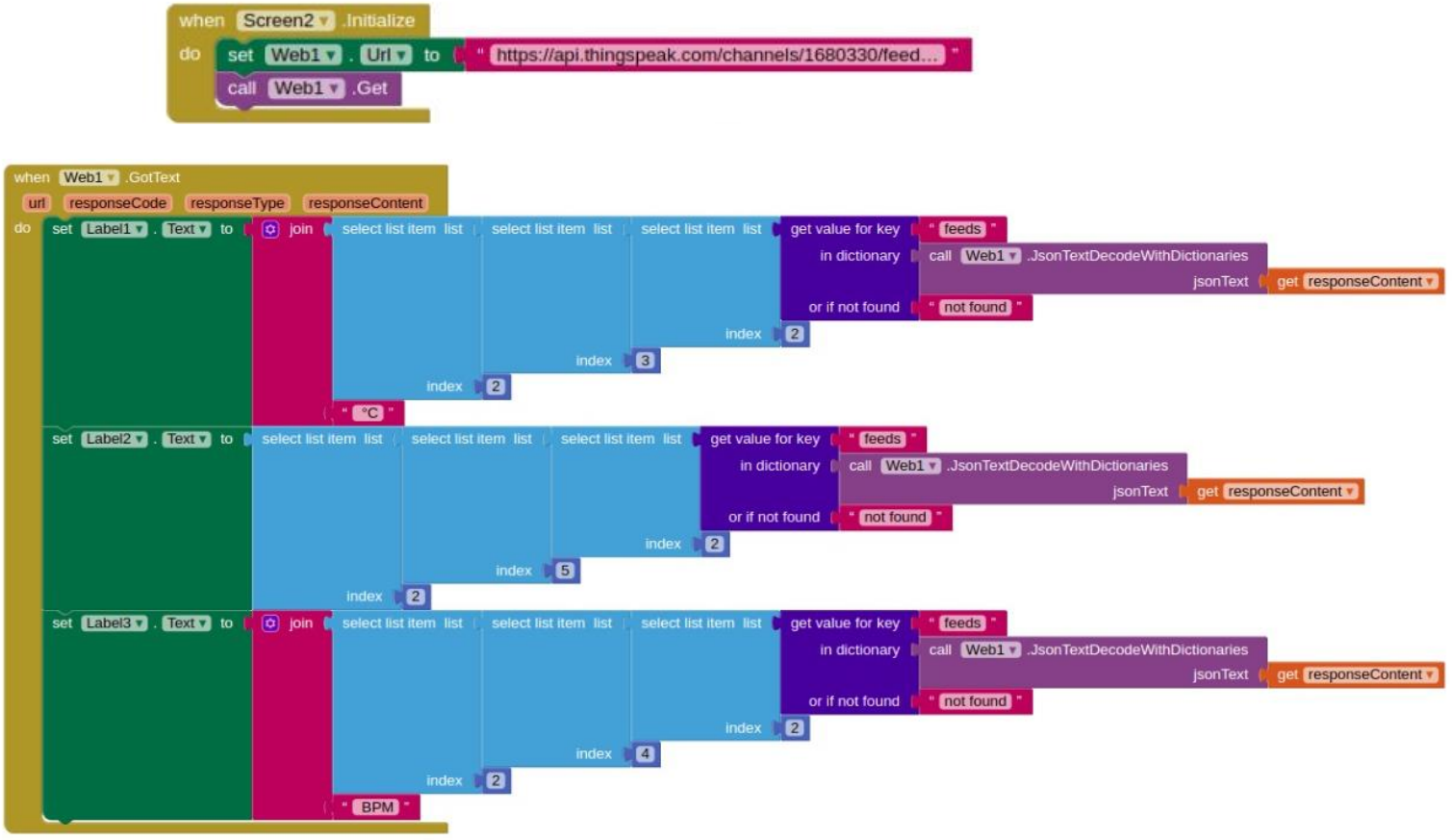
```
123       // Use WiFiClient class to create TCP connections
124     WiFiClient client;
125     const int httpPort = 80;
126     if (!client.connect(host, httpPort)) {
127       Serial.println("Connection failed");
128       return;
129     }
130       // We now create a URI for the request
131     String url = "/trigger/";
132     url += event;
133     url += "/with/key/";
134     url += privateKey;
135     Serial.print("Requesting URL: ");
136     Serial.println(url);
137     // This will send the request to the server
138     client.print(String("GET ") + url + " HTTP/1.1\r\n" +
139                  "Host: " + host + "\r\n" +
140                  "Connection: close\r\n\r\n");
141     while(client.connected())
142     {
143       if(client.available())
144       {
145         String line = client.readStringUntil('\r');
146         Serial.print(line);
147       } else {
148         // No data yet, wait a bit
149         delay(50);
150       };
151     }
152     Serial.println();
153     Serial.println("closing connection");
154     client.stop();
155   }
```

Appendix D

Screen (1) code



Screen (2) Code



Appendix E

File: ECG_Readings.csv

ECG_Readings.csv

	A
1	1.99
2	3.25
3	3.22
4	2.85
.	.
.	.
625	0.94
626	0.74
627	0.36
628	2.45
629	2.03
630	0
631	0
632	1.88
633	0.44

File: ECG_data.m

MATLAB Code

```
function main
Data = GetGoogleSpreadsheet('12OulBwKT82PwDzM8MvKRh5G
data = str2double(Data)
plot(data)
end

function result = GetGoogleSpreadsheet(DOCID)

loginURL = 'https://www.google.com';
csvURL = ['https://docs.google.com/spreadsheet/ccc?ke

%Step 1: go to google.com to collect some cookies
cookieManager = java.net.CookieManager([], java.net.C
java.net.CookieHandler.setDefault(cookieManager);
handler = sun.net.www.protocol.https.Handler;
connection = java.net.URL([],loginURL,handler).openCo
connection.getInputStream();

%Step 2: go to the spreadsheet export url and downloa
connection2 = java.net.URL([],csvURL,handler).openCon
result = connection2.getInputStream();
result = char(readstream(result));

%Step 3: convert the csv to a cell array
result = parseCsv(result)

end

function data = parseCsv(data)
% splits data into individual lines
data = textscan(data,'%s','whitespace','\n');
data = data{1};
end

function out = readstream(inStream)
%READSTREAM Read all bytes from stream to uint8
%From: http://stackoverflow.com/a/1323535

import com.mathworks.mlwidgets.io.InterruptibleStream
byteStream = java.io.ByteArrayOutputStream();
isc = InterruptibleStreamCopier.getInterruptibleStrea
isc.copyStream(inStream, byteStream);
inStream.close();
byteStream.close();
out = typecast(byteStream.toByteArray', 'uint8');

end
```