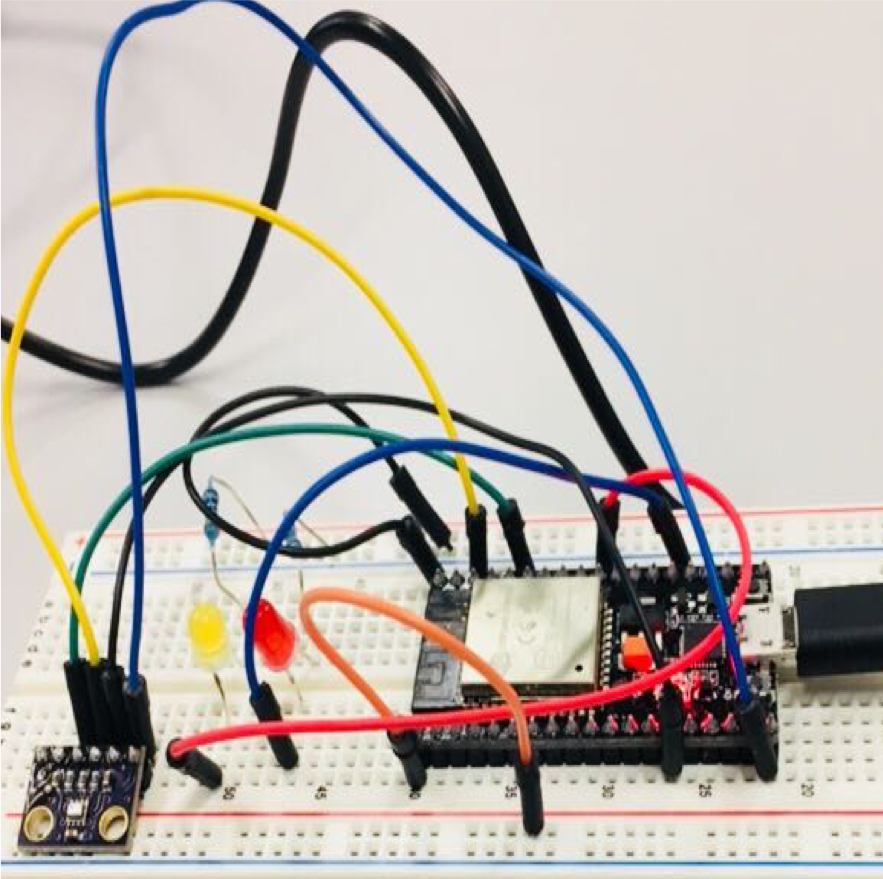
**Offline Monitoring system: An IoT Application**

**CSCI 6838.02 Research Project and Seminar**



**Instructor**

Dr. Pradeep Buddharaju

Assistant Professor

Computer Science

Buddharaju@uhcl.edu

**Mentors**

Thor Arne Lovland, COO

tal@optimesubsea.com

Channa Weeratunge, Senior Software Engineer

cw@optimesubsea.com

**Team Members**

|  |  |
| --- | --- |
| Kamboj, Divya  KambojD4680@UHCL.edu | Rajendran, Priyankaa  [RajendranP6114@UHCL.edu](mailto:RajendranP6114@UHCL.edu) |
| Kasim Razvi, Hajera Nasreen  KasimRazviH3548@UHCL.edu | Sharma, Vidhi Bhaveshkumar  SharmaV5242@UHCL.edu |

**ACKNOWLEGDMENT**

Foremost, we would like to express our sincere gratitude to our mentors, Mr. Thor Arne Lovland and Mr. Channa Weeratunga for their continuous support of our Capstone Project, for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped us throughout the process of conducting this project.

My sincere thanks also goes to the Project Advisor, Dr. Pradeep Buddharaju for his guidance and motivation to consistently pursue the project with excellence.

We also thank the students and Professor of Computer Engineering Dr. Xiaokun Yang, who assisted us with gaining knowledge on the hardware components and their working.

**table of contents**

**acknowledgements** I

**table of contents** ii

**abstract** iii

**1** **INTRODUCTION** 5

1.1 PROBLEM DESCRIPTION 5

1.1.1 PROPOSED SOLUTION 6

1.2 APPLICATION OF THE PROPOSED SYSTEM 6

1.3 IOT BASED APPLICATION 7

1.4 APPLICATION DEVELOPMENT 7

**2** **REQUIREMENTS** 8

2.1BUSINESS REQUIREMENTS 8

2.2 TECHNICAL REQUIREMENTS 8

2.3 DEVICE REQUIREMENTS 9

2.4 FUNCTIONAL REQUIREMENTS 9

2.5 EXTERNAL INTERFACE REQUIREMENTS 9

2.5.1 USER INTERFACES 9

2.5.2 HARDWARE INTERFACES 9

2.5.3 SOFTWARE INTERFACES 9

2.5.4 COMMUNICATION INTERFACES 9

2.6 SYSTEM FEATURES 9

2.7 RESPONSE SEQUENCES 10

2.8 BLUETOOTH SPECIFICATION 10

2.9 POWER MODES FOR ESP 32 10

2.10 SECURITY 10

2.11 WIRELESS CONNECTIVITY 11

2.12 MEMORY 11

**3 SYSTEM ARCHITECTURE** 12

3.1 DEVICE DESCRIPTION 13

3.1.1 ESP32 13

3.1.2 BME280 14

3.1.3 UBLOX NEO6 15

**4** **SYSTEM DESIGN** 17

**5 IMPLEMENTATION** 19

5.1 ARDUINO SETUP 19

5.2 MOBILE APPLICATION DEVELOPMENT PLATFORM 20

5.3 Alternate technologies used 22

5.4 CONCLUSION AND FUTURE WORK 23

**6** **REFERENCES** 23

**ABSTRACT**

The environmental conditions in a closed system are difficult to measure. The closed system is does not have a connection to an external environment to transmit its data in real-time. The Internet of things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and connectivity which enable these objects to connect and exchange data.

Through the use of an ESP32, it is possible to use Bluetooth to connect to the external environment. The preferred external environment shall be a smartphone running on Android or iOS operating system. The project monitors the conditions in closed systems. A closed system is defined as one where the system does not have a connection to an external environment to transmit its data in real-time. For example, monitoring the pressure inside a canister as it floats above the earth’s surface. For such systems, the data should be logged and when possible, downloaded from the system for further processing.

**1. INTRODUCTION**

**1.1 PROBLEM DESCRIPTION**

In spite of the improvement of communication link and despite all progress in communication technologies, there are still very few functioning commercial monitoring systems, which are most off-line and there are still number of issues to deal with. The environmental conditions in a closed system are difficult to measure. A closed system does not have a connection to an external environment to transmit its data in real-time. The project aims to gather real time as well offline data for monitoring and forecasting various conditions and controlling and monitoring the hardware system remotely via Bluetooth or Wi-Fi.

**1.1.1 PROPOSED SOLUTION**

The project scope is to build a system that would monitor the environmental data of a closed system. This would require a microcontroller, sensors, a battery pack and some indicators/switches to provide current status to the user. In addition, an application would be developed to download the logged data and displaying/processing it to the end user. The off-line system as a whole should be light weight, robust, compact and be able to operate with very low power requirements.

The system would be developed using ESP32.The ESP32 chip offers two 32 processing cores which operate at 160MHz, a massive amount of memory, Wi-Fi, Bluetooth and many other features.  We also use the new BME280 sensor which measures the temperature, the humidity, and the barometric pressure. When we power up the project, it connects to the BLE (Bluetooth low energy), and it is going to retrieve the environmental information. The information is stored in external devices such as external SD card .The readings are updated every few seconds and the weather forecast every hour. In this project we use the latest technologies available to a maker today.The system would be an application of IoT. The Internet of things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and connectivity which enable these objects to connect and exchange data.

**1.2 APPLICATION OF THE PROPOSED SYSTEM**

The proposed system will be used to monitor the temperature and pressure inside the vessels, which are placed deep inside the sea. The data will be recorded and stored. The stored data can be viewed or downloaded via a mobile application [8].

Such a system can also be used in various other fields such as weather monitoring and agriculture.

* **Weather systems** can be used designed to collect data from various vehicles on the road, vehicles moving on the road will wirelessly communicate the weather and road condition data that is inclusive of air temperature, barometric pressure, visibility or light, motion and other data needed. This data helps to build more accurate forecast and provide flexible real time monitoring at different time horizon.
* **IOT weather reporting system has application to farmers as well**. The weather forecasting plays very important role in the field of agriculture. Agricultural process i.e. preparation of soil, sowing, irrigation, harvesting and storage of crops is directly dependent on weather condition leaving farmers vulnerable to weather hazards.
* **Smart Lighting** Intelligent and weather adaptive lighting in street lights.
* **Smart Roads** Intelligent Highways with warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.
* **Green Houses** Control micro-climate conditions to maximize the production of fruits and vegetables and its quality.

**1.3 IOT BASED APPLICATION**

The product developed is an IoT based application that can be modified and used in many other application areas. The hardware developed is used to measure environmental conditions. It also includes GPS for location.

A mobile application can be used to control the hardware remotely and data can be stored or downloaded via mobile application for data analytics.

The product can be modified by adding features such as motion sensors to develop **“Smart baby monitoring”** system to monitor the movements of baby, especially while the baby is sleeping. A mobile application can be used to control and monitor the environment around the baby, like controlling temperature of room, playing soothing music, controlling the lights etc.

**1.4 APPLICATION DEVELOPMENT**

The system uses Arduino IDE to program the ESP32 microcontroller which has inbuilt Bluetooth and WIFI features. The feature is not classic Bluetooth which instead uses Bluetooth low energy (BLE) which when compared to classic Bluetooth provides reduced power consumption and cost while communicating with an external device [6]. The BLE is used to communicate with the mobile device using Generic Attribute Profile (GATT) to transfer data back and forth using services and characteristics. GATT is built over the Attribute profile (ATT) for Bluetooth low energy implementations. Its role within the ESP32 (server) is to define an exposed data structure which the mobile device (client) can use to understand and reference for data access purposes [7]. GATT includes the following:

* **Profile** is a virtual grouping of one or more services. It is not visible to Clients when discovering the GATT of a server nor exist on the server. Standard profiles are defined to guarantee interoperability between applications of different devices, If Temperature, pressure profile is used (GATT specifications) so that other devices that support temperature or pressure can interface with the device (Mobile Application or any other device) with no compatibility issues.
* **Services** are collections of characteristics and associated behaviors to accomplish a specific functionality. Services are assigned unique identifiers called UUIDs. Bluetooth adopted services have 16-bit UUIDs assigned, whereas user-defined get assigned 128-bit UUIDs. Service UUID in the system used is **6E400001-B5A3-F393-E0A9-E50E24DCCA9E.**
* **Characteristics** consists of a single value and information about the value. The information include configuration information about how the value is accessed and represented in the ESP32 and mobile application. It defines properties that define what operations are allowed or can be configured for the value. Some operations used in the system:
  + - **Read:** Mobile application reads the value when read is enabled.
    - **Write:** Lets the mobile application to change the value when enabled.

Thunkable MIT’s App Inventor is the platform in which the mobile application for android was developed. The system connects to the microcontroller with Bluetooth and data communication is done to retrieve the pressure/temperature values.

An external device SD card is used to store the data when the user is offline. ESP 32 is connected to SD card and data communication is done using SPI protocol. The values are stored in an SD card as text file format.

**2. REQUIREMENTS**

* 1. **BUSINESS REQUIREMENTS**
* The main purpose of the project is to design and implement the offline monitoring system with sensors and GPS which aids in retrieving the data from the closed system.
* This project will consist of ESP 32 microcontroller, BME280, and UBLOX NEO6 to be connected with the microcontroller and modules include programming the microcontroller, connecting the system with a mobile device to retrieve the data that is being monitored.
* The company and the employees will be benefited from the offline monitoring of the environment.

**2.2 TECHNICAL REQUIREMENTS**

1. Device connection
2. C programming for configuration
3. Mobile application development
4. Test cases
   1. **DEVICE REQUIREMENTS**
5. ESP 32 development board
6. BME 280 Sensor
7. NEO-6 (ublox 6 GPS Modules)
8. Breadboard
9. Jumper wires
10. 3 in 1 wires
11. SD card Adaptor
12. SD card – 8 GB
    1. **FUNCTIONAL REQUIREMENTS**

The pressure/temperature sensors, GPS provide the input signal for the microcontroller to store and process it for the users to view the status of the medium at that point.

* 1. **EXTERNAL INTERFACE REQUIREMENTS**

**2.5.1 USER INTERFACES**

The user interfaces involve different views such as line graph, bar chart, and other views. The user has the ability to start and end logging, to erase the previous log files, set the maximum temperature/pressure limits for viewing the graph.

**2.5.2 HARDWARE INTERFACES**

The devices such as ESP32, BME280, Ublox, Battery, Breadboard, Wires, LEDs, Resistors.

**2.5.3 SOFTWARE INTERFACES**

The software used is Arduino and MIT app inventor Thunkable.

**2.5.4 COMMUNICATION INTERFACES**

The communication protocol UART/SPI is used to send information.

* 1. **SYSTEM FEATURES**

The features include sensing the input signal from the pressure/temperature sensor and GPS, converting the analog input signal to digital signal, retrieve the data from the microcontroller in log file format and the information is displayed in a tabular/graphical format on the mobile app device.

* 1. **RESPONSE SEQUENCES**

The user actions involve retrieving the data for a given time-in and time-out. The system starts/ends logging the data from the microcontroller as per the command received. The initial threshold of data logging is set to be 60 minutes, and the logging interval for which the data is fetched is 10 secs.

* 1. **BLUETOOTH SPECIFICATION**

The Bluetooth stack of ESP32 is compliant with Bluetooth v4.2 BR / EDR and BLE specification.

* 1. **POWER MODES FOR ESP 32**
  + **Active mode:** The chip radio is powered on. The chip can receive, transmit, or listen.
  + **Modem-sleep mode**: The CPU is operational, and the clock is configurable. The Wi-Fi/Bluetooth baseband and radio are disabled.
  + **Light-sleep mode:** The CPU is paused. The RTC memory and RTC peripherals, as well as the ULP co-processor are running. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
  + **Deep-sleep mode:** Only RTC memory and RTC peripherals are powered on. Wi-Fi and Bluetooth connection data are stored in RTC memory. The ULP co-processor can work.
  + **Hibernation mode:** The internal 8-MHz oscillator and ULP co-processor are disabled. The RTC recovery memory is powered down. Only one RTC timer on the slow clock and some RTC GPIOs are active. The RTC timer or the RTC GPIOs can wake up the chip from the Hibernation mode.

**2.10 SECURITY**

* IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
* Secure boot
* Flash encryption
* 1024-bit OTP, up to 768-bit for customers
* Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)
  1. **WIRELESS CONNECTIVITY**
* Wi-Fi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz up to 150 Mbit/s)
* Bluetooth: v4.2 BR/EDR and Bluetooth Low Energy (BLE)
  1. **MEMORY**
* Internal memory
  + - ROM: 448 KiB
* For booting and core functions
  + - SRAM: 520 KiB
* For data and instruction
  + - RTC slow SRAM: 8 KiB
* For co-processor accessing during deep-sleep mode.
  + - RTC fast SRAM: 8 KiB
* For data storage and main CPU during RTC Boot from the deep-sleep mode**.**
  + - EFuse: 1 Kibit of which 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including Flash-Encryption and Chip-ID.
* Embedded flash connected internally via IO16, IO17, SD\_CMD, SD\_CLK, SD\_DATA\_0 and SD\_DATA\_1 on ESP32-D2WD and ESP32-PICO-D4.
  + - 0 MiB (ESP32-D0WDQ6, ESP32-D0WD, and ESP32-S0WD chips)
    - 2 MiB (ESP32-D2WD chip)
    - 4 MiB (ESP32-PICO-D4 SiP module)
* External flash & SRAM ESP32 supports up to four 16 MiB external QSPI flashes and SRAMs with hardware encryption based on AES to protect developers' programs and data. ESP32 can access the external QSPI flash and SRAM through high-speed caches. Up to 16 MiB of external flash are memory-mapped onto the CPU code space, supporting 8-bit, 16-bit and 32-bit access. Code execution is supported. Up to 8 MiB of external flash/SRAM memory are mapped onto the CPU data space, supporting 8-bit, 16-bit and 32-bit access. Data-read is supported on the flash and SRAM. Data-write is supported on the SRAM.

**3. SYSTEM ARCHITECTURE**

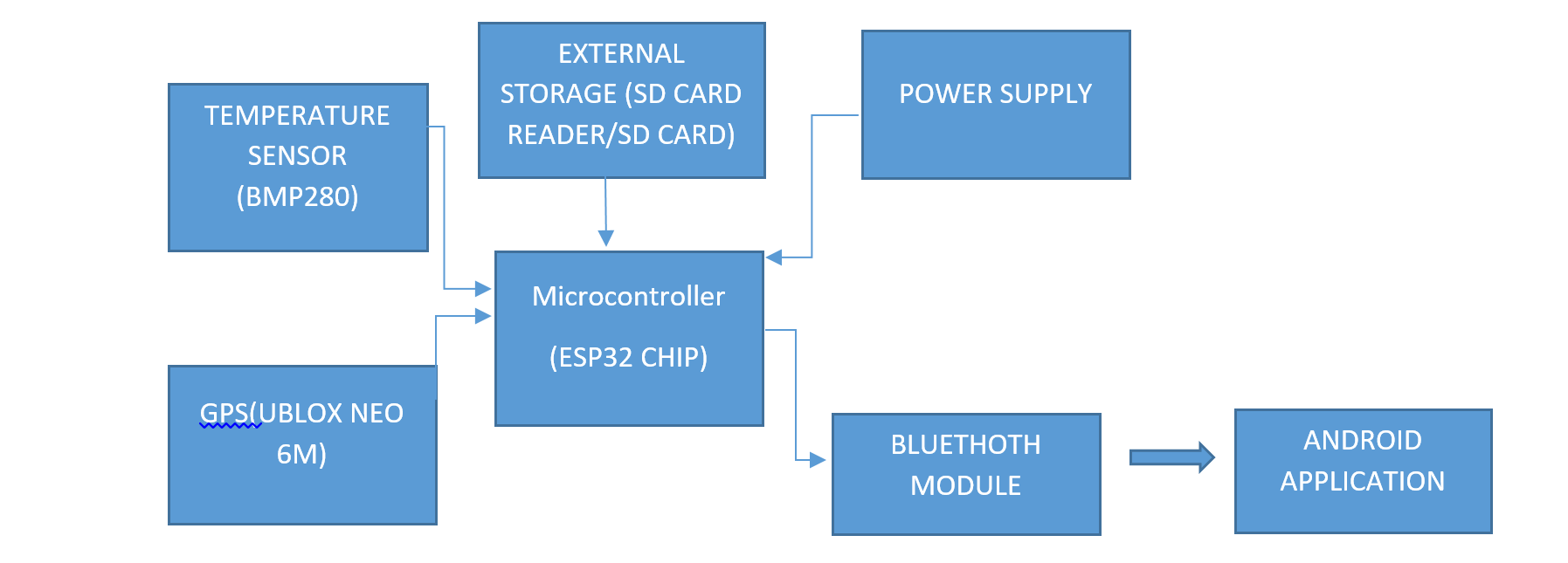


Fig-1: System Flow

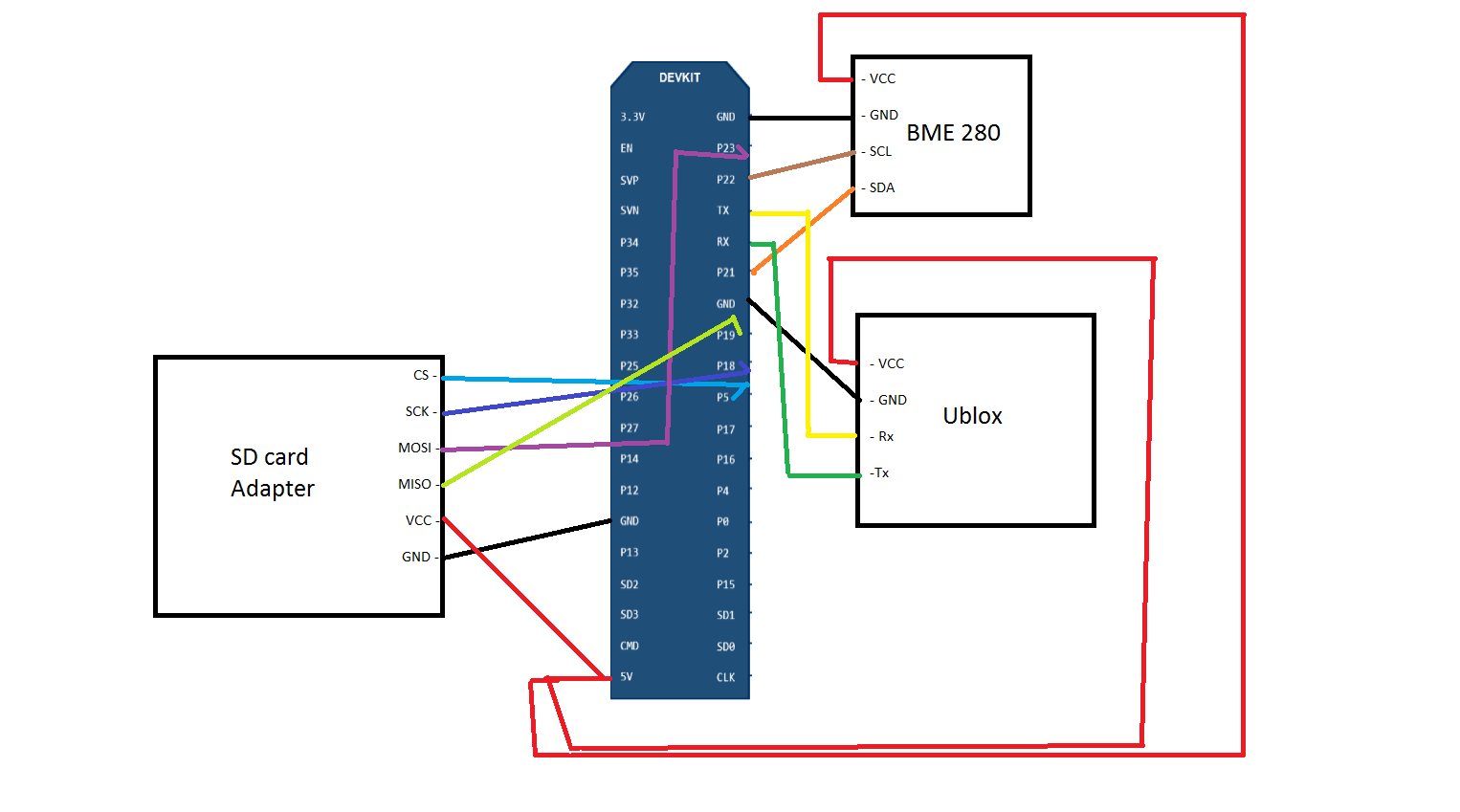
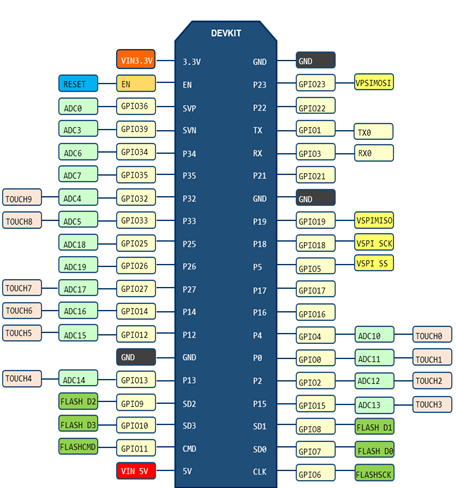
****

Fig-2: Pin Diagram of SD card, ESP 32 and Sensors

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BME 280** | **ESP 32** | **Ublox** | **ESP 32** | **SD Card Adapter** | **ESP 32** |
| VCC | 5 V | GND | GND | CS | GPIO 5 |
| GND | GND | V | 5 V | SCK | GPIO 18 |
| SCL | GPIO 22 | Rx | GPIO 3 | MOSI | GPIO 23 |
| SDA | GPIO 21 | Tx | GPIO 1 | MISO | GPIO 19 |
|  |  |  |  | VCC | 5 V |
|  |  |  |  | GND | GND |

Fig-3: Pin Connection for ESP 32 with BME 280 and Ublox

* 1. **DEVICE DESCRIPTION**
  2. **1 ESP32**

ESP32 [1]  is a successor to the ESP8266 microcontroller .ESP32 is a series of low cost, low power system on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth with TSMC ultra-low-power 40 nm technology.ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process. ESP32 uses CMOS for single-chip fully-integrated radio and baseband, and also integrates advanced calibration circuitries that allow the solution to dynamically adjust itself and remove external circuit imperfections or adjust to changes in external conditions [1].It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and different power profiles. ESP32 contains an antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules.

**FEATURES OF ESP32**

**PROCESSORS**

* CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS
* Ultra-low power (ULP) co-processor

**WIRELESS CONNECTIVITY**

* + Wi-Fi: 802.11 b/g/n
  + Bluetooth: v4.2 BR/EDR and BLE

**MEMORY** 520 KiB SRAM

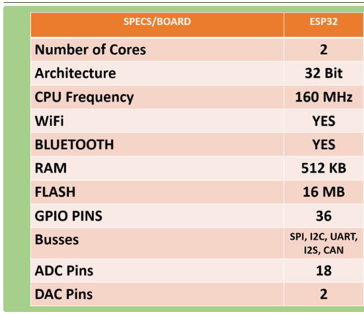


Fig-4: ESP 32 specifications [3]

* + 1. **BME280**

The BME280 is an environmental sensor developed specially for low power devices. The device provides high accuracy and linearity for pressure and temperature sensing. The sensor provides low current consumption (3.6 μA @1Hz), high robustness and long term stability.

The humidity sensor offers high performance rate that benefits emerging applications such as context awareness, and high accuracy over a wide temperature range. The pressure sensor is an absolute barometric pressure sensor with features exceptionally high accuracy and resolution at very low noise. The temperature sensor provides low noise and high resolution to measure ambient temperature.

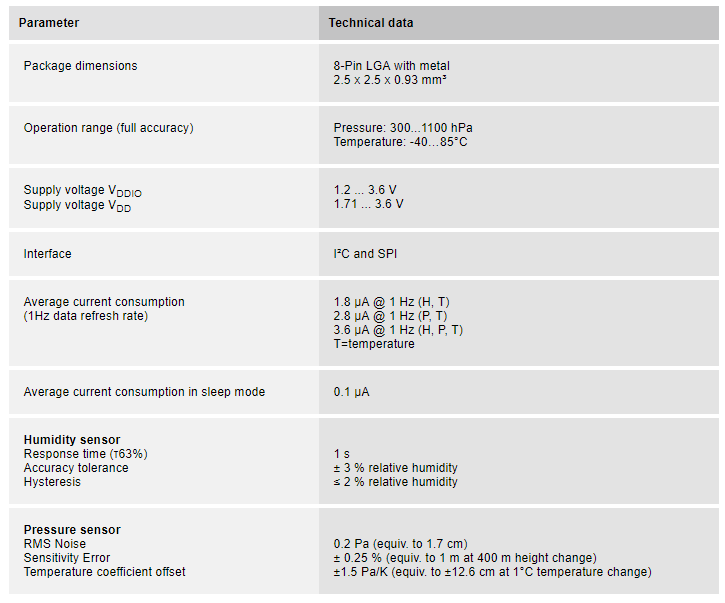


Fig-5: BME 280 specifications [4]

**3.1.3 UBLOX NEO6**

The NEO-6 module series is a GPS receivers offering the high-performance ublox 6 positioning engine. The GPS Receiver offers cost effective and flexible connectivity options in a miniature 16 x 12.2 x 2.4 mm package. The low power requirement of the GS module make them suitable for mobile devices having low power and space constraints.

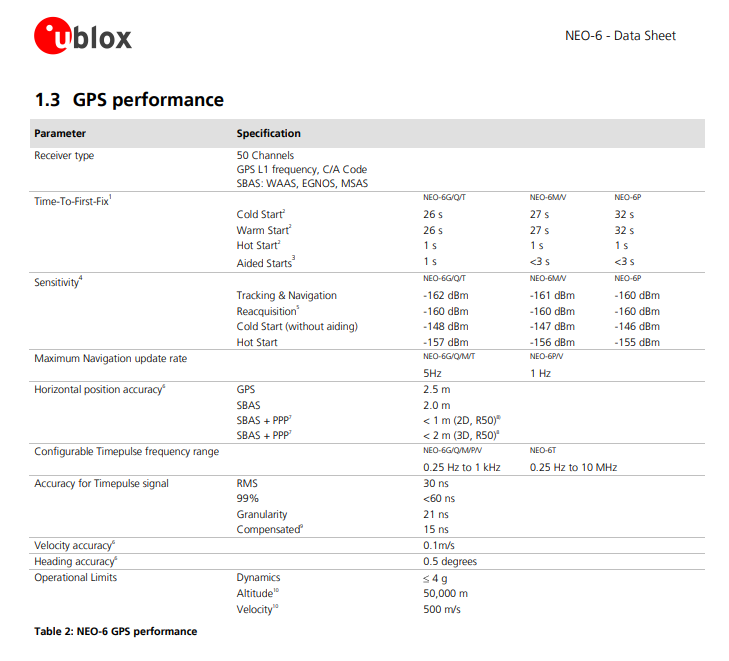


Fig-6: Ublox specifications [5]

1. **SYSTEM DESIGN**

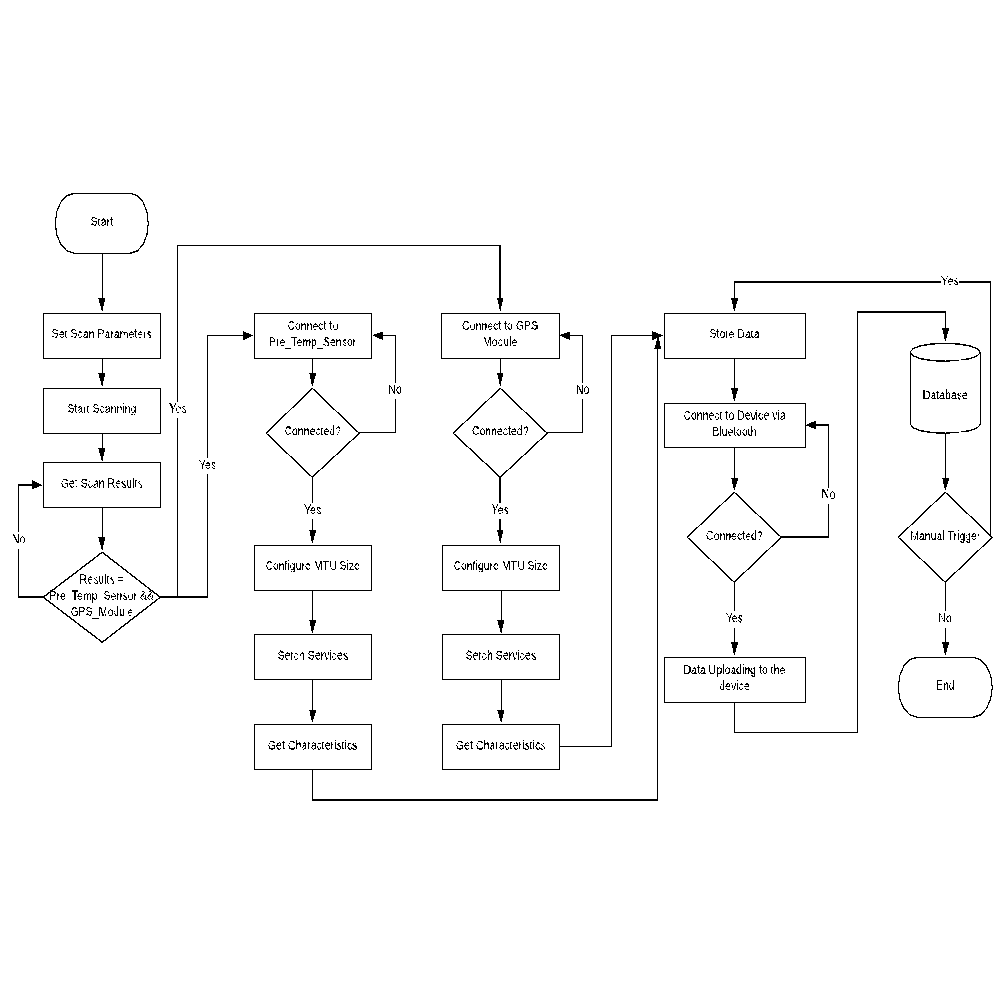
****

Fig-8: Flow chart of ESP 32 and BME280

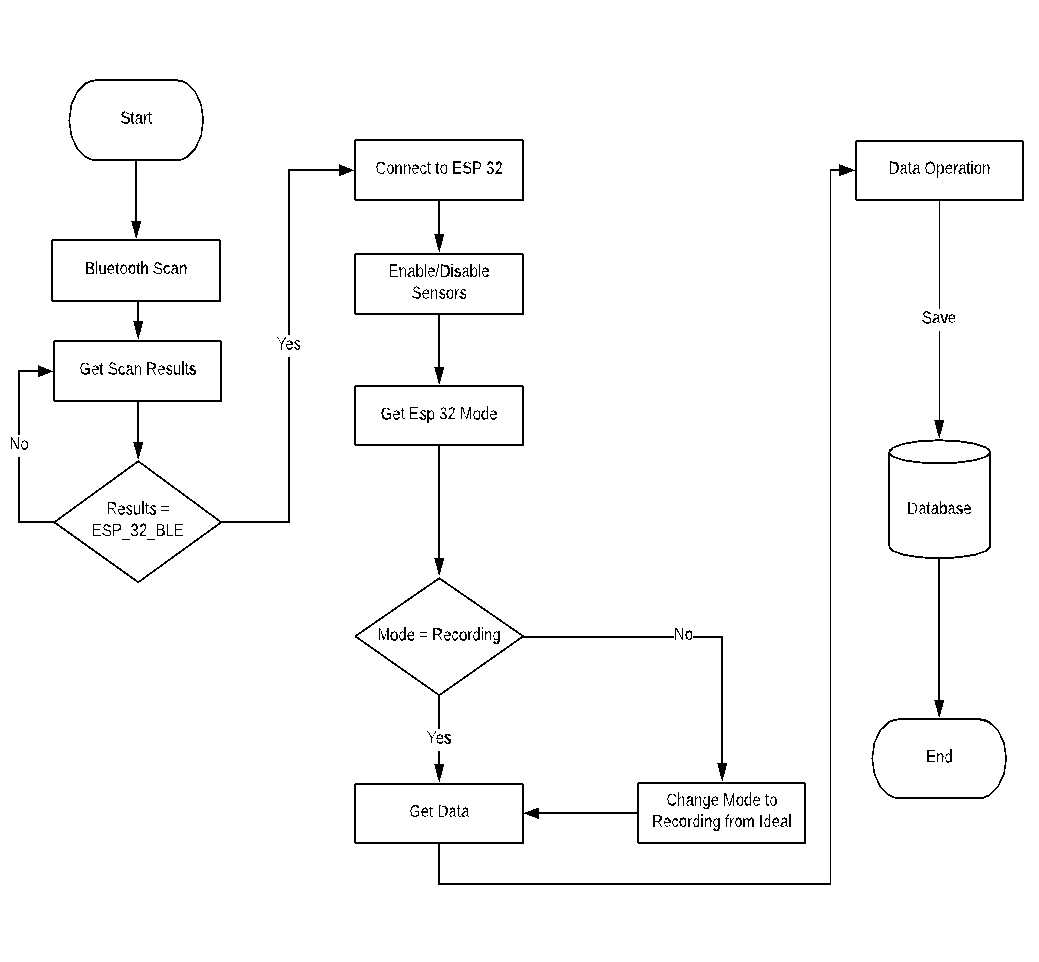


Fig-9: Flow chart of ESP 32 and Mobile device

1. **IMPLEMENTATION**
   1. **ARDUINO SETUP:**

Arduino [3] is an open-source platform used for building electronics projects. Arduino consists of a hardware device and a software connected to it for programming the hardware. The software can be used to program any hardware device (microcontroller).A simple USB cable can be used to load the program to the hardware device. The Arduino IDE uses a simplified version of C++, making it easier to learn to program.

**ESP32 Package Installation**

The ESP32 package can be easily installed in Arduino IDE. For Windows, instructions tell you to open "Git GUI" you have to download and set up "Git”. Once you enter the details, one can easily download the ESP32 package and ready to program using Arduino.

**ESP32 BLE Example Sketch**

In Arduino IDE [4] the first thing to do is, go to Tools / Board and select the appropriate board. Chose "ESP32 Dev Module" for my board. Also choose the correct COM port after connecting the board to your computer via the USB cable.

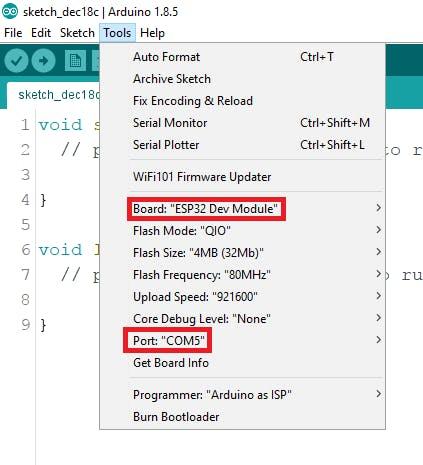


Fig-10: Arduino Example

In order to check if the ESP32 installation go to File / Examples / ESP32 BLE Arduino and you should see several example sketches, like "BLE\_scan", "BLE\_notify", etc. This means everything is set up properly in Arduino IDE!

**5.2** **MOBILE APPLICATION DEVELOPMENT PLATFORM**

**Thunkable Application development**

For the Android application development we used MIT Thunkable App [2].Thunkable is a visual app-building tool for Android and iOS. We are developing android application and not the IOS application as it does not support Bluetooth support yet.

First go to the Thunkable site and set up an account or log in with a Google account. If you're new to Thunkable you won't see any existing project.

Click "Apps" at the top left and click "Upload app project (.aia) from your computer". The "native" file type for Thunkable is ".aia" files and these files will allow you to view and edit code blocks within Thunkable.

Once you to the app's home screen where you can edit the user interface. To view and edit the code blocks, click "Blocks" sort of at the top left, next to "Designer".

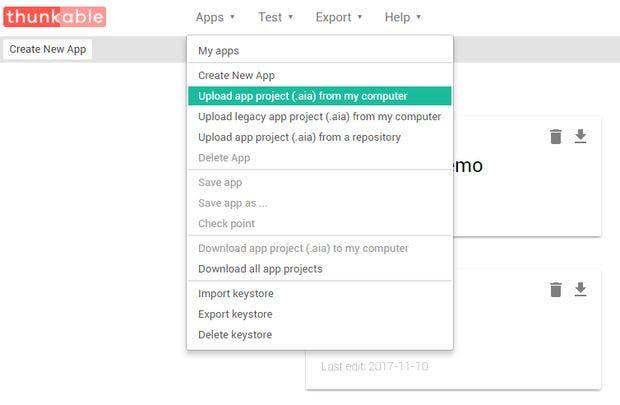


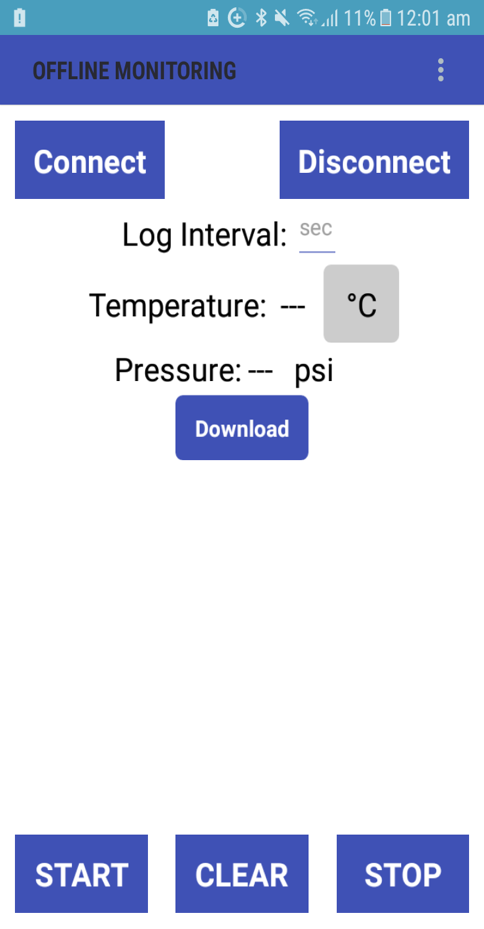
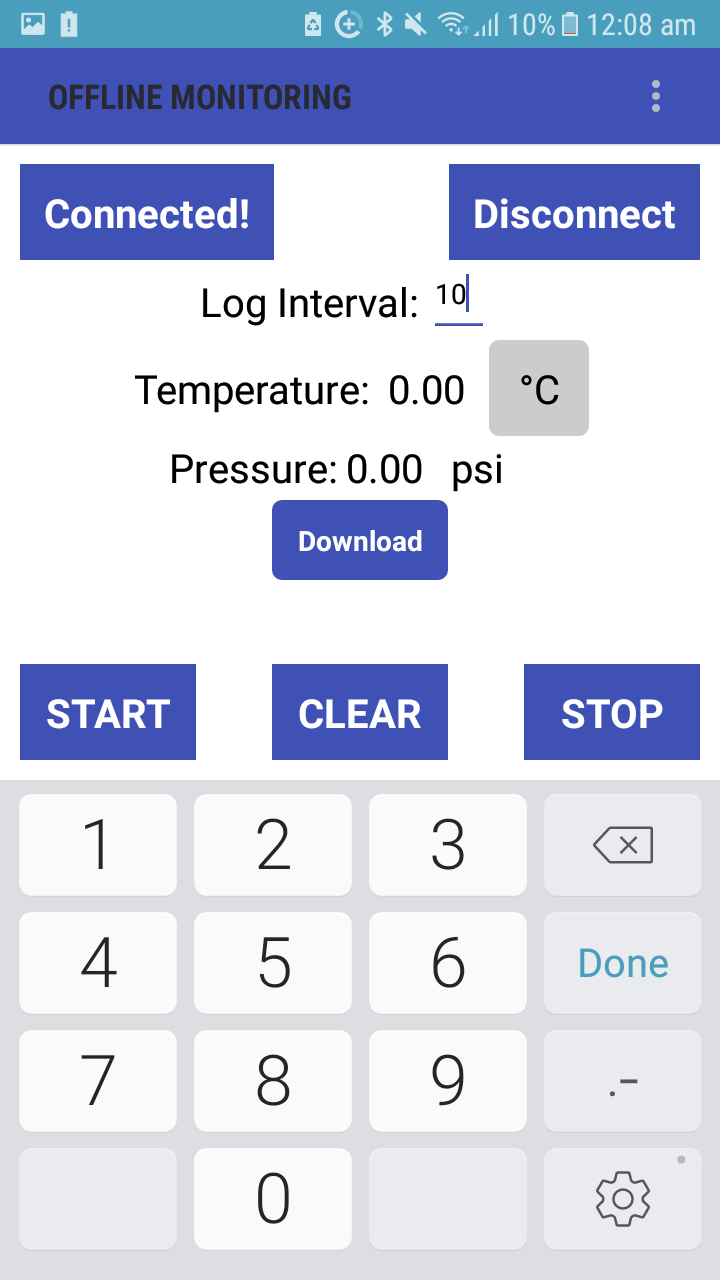
Fig-11: Thunkable Application Creation

**Thunkable Companion App**

In order to do live testing of the developed application, we can use the Thunkable companion app on your mobile device. The application can be tested without compiling and downloading it every time. We can simply install it on your mobile device and under the "Test" tab at the top click "Thunkable Live" and it will bring up a QR code on the screen. Open the Thunkable app on your mobile device and scan the QR code to live test.

To get the app on your phone, simply click "Export" and "App (provide QR code for .apk)" and scan the QR code with your phone using the Thunkable app. You can then install the app and open it. When you first open the app, it will ask you to turn on Bluetooth if you haven't already and click "Yes". When the app is connected to your ESP32 it will print out arbitrary values that are sent to it from the ESP32.

**Offline Monitoring System Mobile App:**

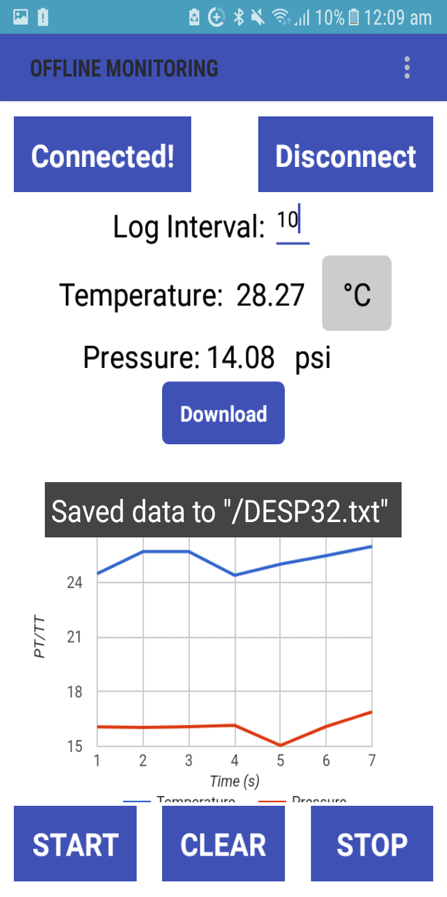
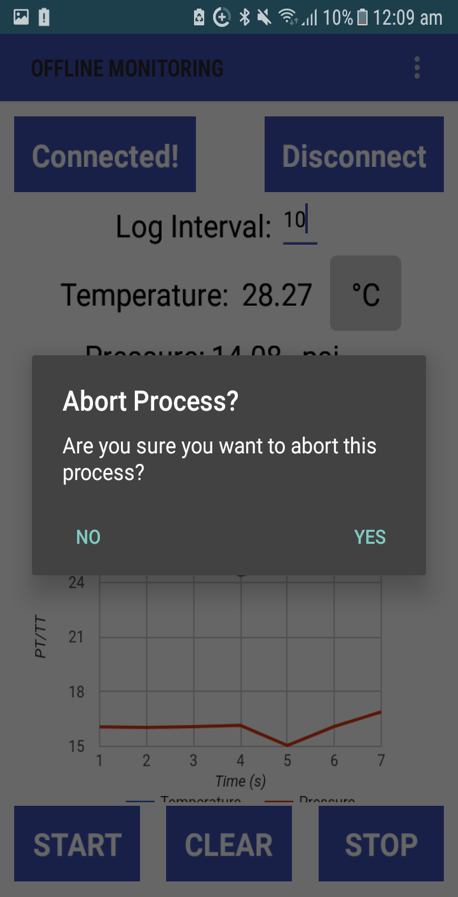
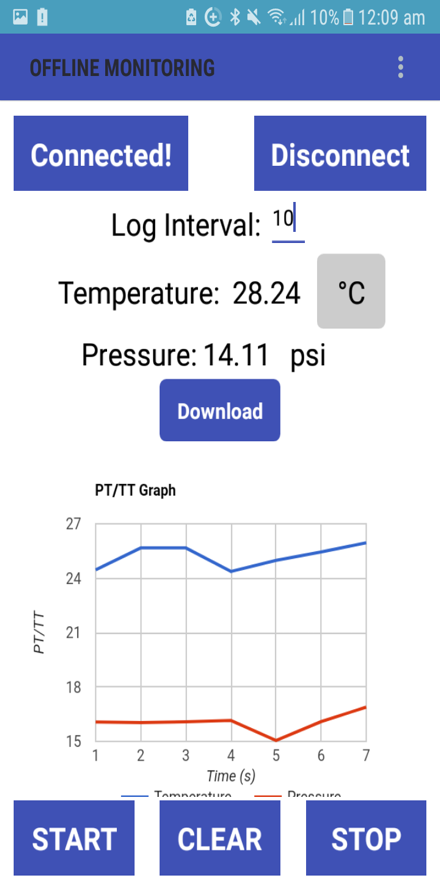


Fig-12: Thunkable Mobile Application Screenshots

**5.3 ALTERNATE TECHNOLOGIES USED**

**Android studio/BLE Intro**

Android 4.3 (API level 18) introduces Bluetooth low energy platform[6] .The build-in platform for BLE provides API that can be used for application development and build services such as transmit information, discover devices and query for services.

**Common uses of BLE are:**

To transfer small amounts of data between nearby devices.

To interact with sensors giving end users a customized experience based on their current location.

Bluetooth Low Energy (BLE) are designed for low power devices such as proximity sensors, heart rate monitors, fitness devices and temperature sensors.

**Details of BLE:**

**Generic Attribute Profile (GATT)**—The GATT is a generalized specification in order to send and receive small amount of data known as attributes over BLE link. All the currently available low power devices are based on GATT.

The Bluetooth SIG has many profiles for Low Energy devices. A profile is a specification for how a device works in a particular application.

**Attribute Protocol (ATT)**—ATT is specially designed to run on BLE devices. Each attribute has a Universally Unique Identifier (UUID) having 128 bit format .The attributes used by ATT are formatted as characteristics and services. GATT is built on top of the Attribute Protocol (ATT) also referred to as GATT/ATT.

**Characteristic**—Characteristic can be seen as a class having single value and 0-n descriptors that describe the characteristic's value.

**Descriptor**—Descriptors are the attributes that describe a characteristic value.

**Service**—A service is a collection of characteristics. For example, suppose we have a service called "Heart Rate Monitor" that includes characteristics such as "heart rate measurement."

**5.4 CONCLUSIONS AND FUTURE WORK**

The project is an IoT system, where an ESP32 microcontroller is used along with the BME280 pressure/temperature sensors. The sensors send the pressure/temperature values to the ESP32 which is collected and stored in an external device Micro SD card in offline mode. The end user can view the stored values anytime in a mobile device which will be connected to the ESP32 through Bluetooth low energy connection. The real time values are viewed in graph or downloaded as a text file when stopped whereas offline values are retrieved from a SD card.

**FUTURE WORK**

* Addition of motion sensors, LED lights control
* Use of Wi-Fi And Cloud AWS (for data storage/retrieval in cloud)
* Online Data Monitoring
* Data Analysis (to detect patterns for generating alarms and other useful information from stored data)

**6. REFERENCES**

[1]ESP32Overview|EspressifSystems, www.espressif.com/en/products/hardware/esp32/overview. “ESP32 WeatherStation.” educ8s.Tv - Watch Learn Build, 4 Apr. 2018, educ8s.tv/esp32-weather-station.

[2] “Thunkable.com: Drag and Drop Mobile App Builder for IOS and Android.” Thunkable.com: Drag and Drop Mobile App Builder for IOS and Android, thunkable.com

[3] Arduino - Tutorials. Available: https://www.arduino.cc/en/Tutorial/HomePage.

[4] “Build Your First App,” Android Developers, 22-Feb-2018. [Online]. Available: https://developer.android.com/training/basics/firstapp/index.html.

[5] Instructables. “ESP32 WiFi Weather Station With a BME280 Sensor.” Instructables.com, Instructables, 21 Nov. 2017, www.instructables.com/id/ESP32-WiFi-Weather-Station-With-a-BME280-Sensor/

[6] “Bluetooth Low Energy.” Android Developers, 9 Mar. 2018, developer.android.com/guide/topics/connectivity/bluetooth-le.html.

[7] “Bluetooth GATT: How to Design Custom Services & Characteristics,” Novel Bits, 04-Sep-2017. [Online]. Available: http://www.novelbits.io/bluetooth-gatt-services-characteristics/. [Accessed: 25-Apr-2018].

[8]“The 10 Most Popular Internet of Things Applications Right Now.” IoT Analytics, iot-analytics.com/10-internet-of-things-applications/.