Department of Electrical Engineering and Computer Science - EECS3215: Embedded Systems

Baby Monitoring System Project Report

Dr. Ebrahim Ghafar-Zadeh

Submitted by:

Rishahb Bhatnagar (213671714)
Eun Diana Lee (212475638)
Ekram Bhuiyan (212914479)

1. ABSTRACT

The baby monitor project is a Wi-Fi enabled system that measures the changes in pressure on a newly devised fabric sensor, which detects changes in voltage. The system design was developed in partnership with Professor Ebhrahim Ghafar-Zadeh and his research group. The overall system design combined our knowledge in electrical circuits, software and Internet of Things (IoT). The purpose and ultimate goal of this project was to develop a mechanism that notifies a user if the movement of a baby is detected. This goal was successfully achieved while allowing the students to gain a deep practical experience in Embedded Systems design.

2. INTRODUCTION

Recent advances in sensor technology have led to new innovations in how embedded systems can be used for baby monitoring systems. For this project, a conductive fabric sensor, one of the latest inventions, was used to monitor vital signs or other electrical impulses of a subject - the subject in this case being babies, especially the newly born. The sensor is woven or knitted from conductive fibers; and, when in contact with the body of a subject, it receives signals from the subject and transmits them to an embedded device. The embedded device then processes and passes the information to a web server, which presents the information as an aesthetic user interface. Thus, the sensor ought to provide uni-directional communication by monitoring the subject’s movements.

3. EQUIPMENT

Hardware

- ESP8266 Huzzah feather Development Board (Shown in Figure 1)
- Baby Monitoring Fabric (25 cm x 5 cm)
- 1 Bread Board
- 3 Jumper Cables
- Tektronix Oscilloscope
- 1 BNC cable
- 3D Printed Fabric Stand
- Copper wiring (Length 25 cm)
- Baby Monitoring Stand (See System Diagram – Low level 1 and 2) – We have two diagrams because at one point, it was required that we switch stands.

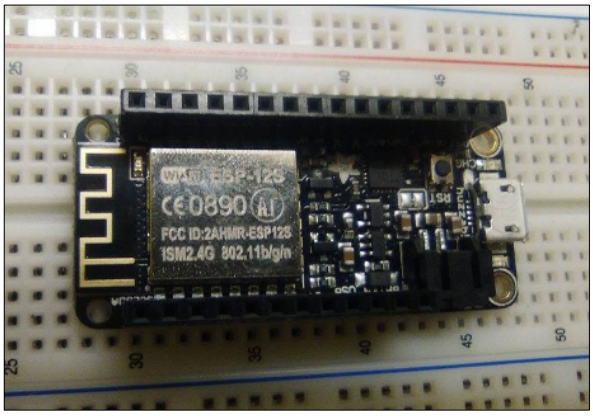

Figure 1 – ESP8266 Huzzah; *The Huzzah development board is the embedded system we chose after a previous one failed (See Issues encountered)*

4. PROCEDURE & METHODS

The high-level view of the project is shown in Figure 2A; the detailed view of each component is also shown in Figure 2B and Figure 2C. In this project, a trial-and-error method was taken since not much is known about the provided conductive fabric and its behavior. The first step to create the device based on the diagram is to use an arbitrary function generator – in this project, an oscilloscope is used – in order to apply the AC power source by generating a sine wave across the fabric. In this project, a sine wave with a frequency of 25 Hz (See System Design) and a Vpp of 750 mV (See Issues encountered) were applied. Then, this fabric is fixed using the 3D-printed fabric stands, which is shown in Figure 5 and Figure 6. Initially, the first fabric stand shown in Figure 5 was used; however, due to a minor issue, the second fabric stand shown in Figure 6 was used for the project. With the first fabric stand, since its mounts – identified with blue circles in Figure 5 – were not as good as expected, a set of jumper cables were used directly to connect the fabric with the oscilloscope and the main board; the issue with the mounts was further discussed in the Issues Encountered section. With the second fabric holder, the jumper cables were connected directly to its contact mounts since the connection between the mount and the fabric was much better than the first one. After that, the other side of the jumper cables was directly attached to the analog-to-digital converter (ADC) pin of the Huzzah development board. Once connected, the software would garner data from the ADC pin and then using MQTT protocol, transmit the collected data to the Adafruit servers. The information collected will then be aesthetically displayed as a graph via Adafruit's interface. The source code of the software can be found in Appendix.

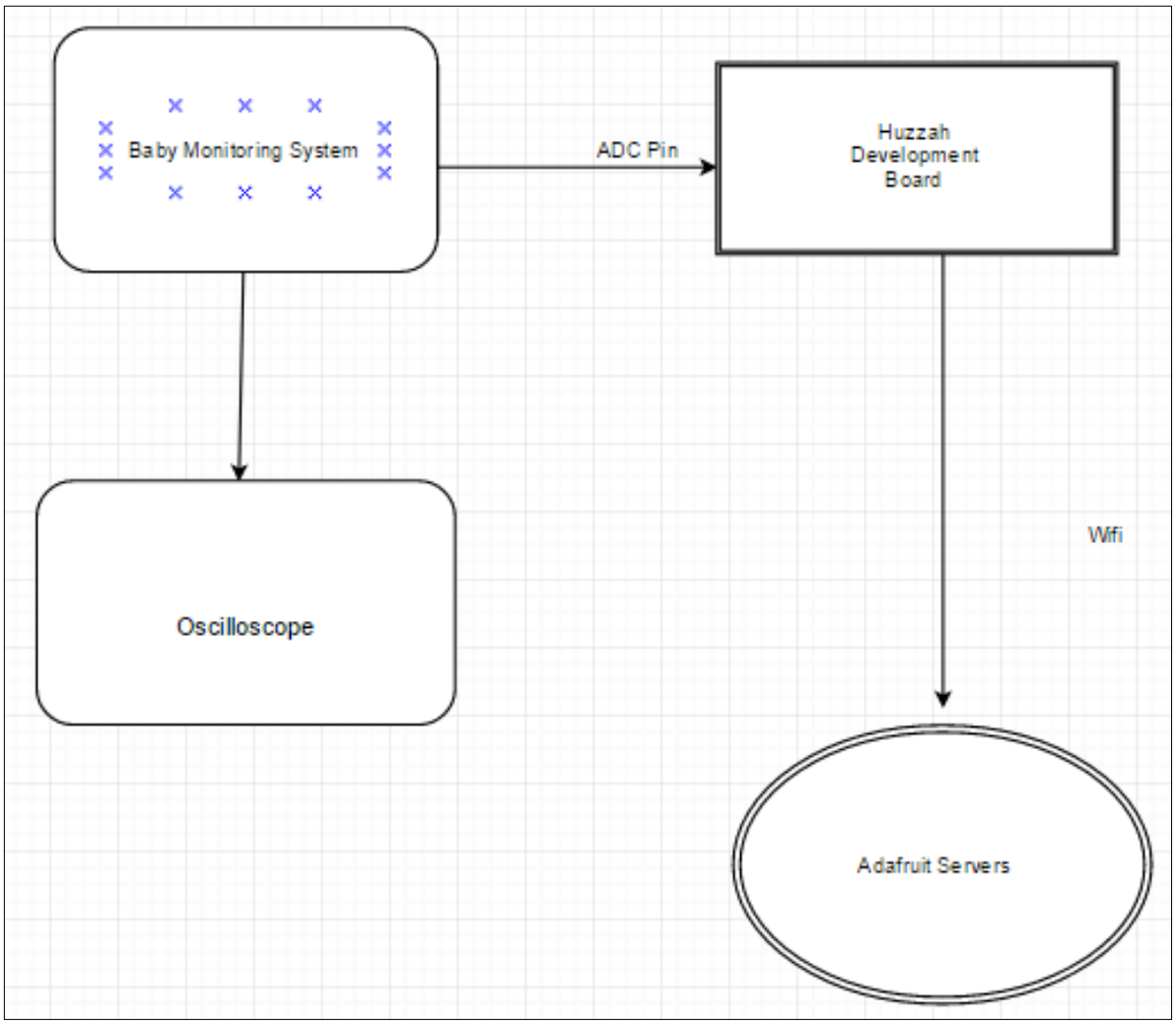

Figure 2.a - The Diagram of the System (High Level)

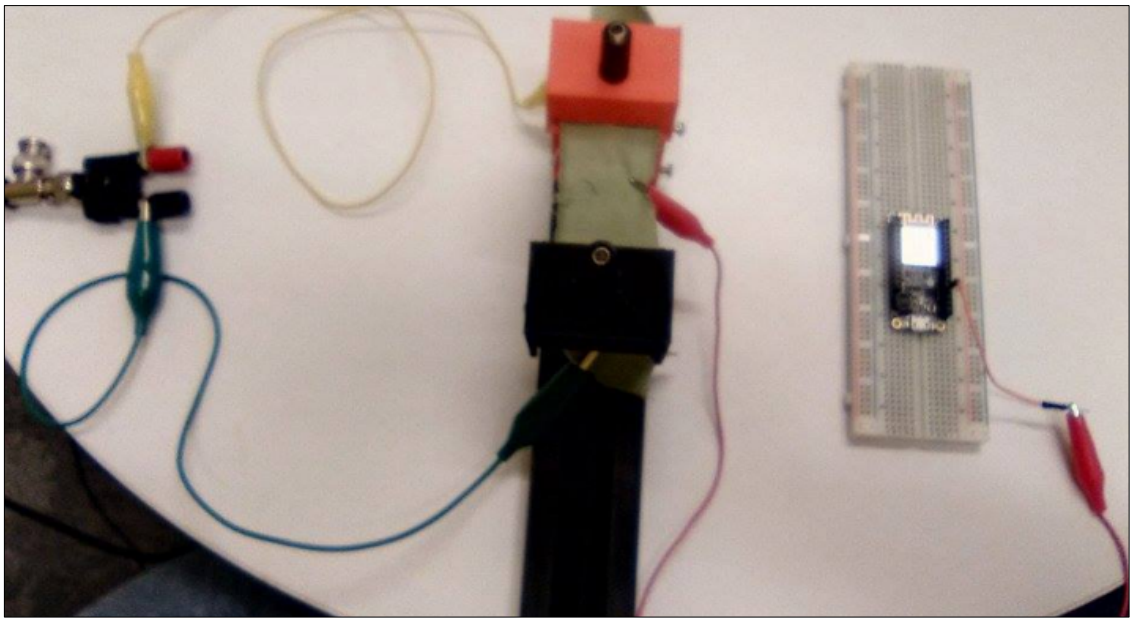

Figure 2.b - System Diagram (Low Level 1)

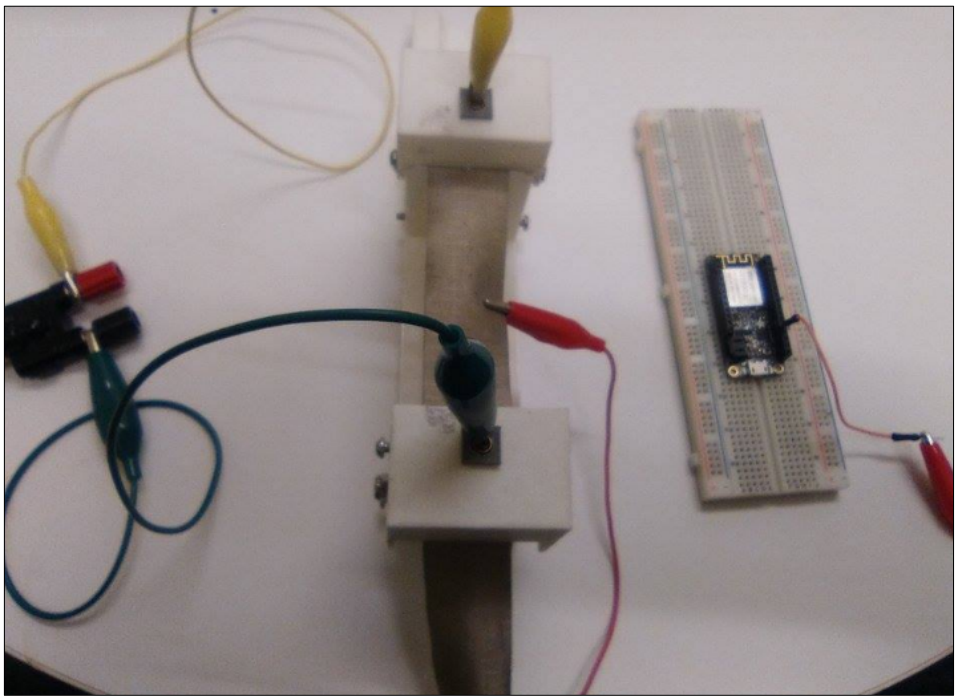

Figure 2.c - System Diagram (Low Level 2)

5. RESULT

Prior to discussing about the obtained results, it is important to note that the pressure is applied using a finger wrapped in non-conductive materials. This result section is divided into two integral parts: 1. The result displayed on the oscilloscope that confirms that an applied pressure on the fabric does affect the voltage drop across it (confirming the useful property of the fabric) and 2. The result obtained via web app following a reading from the ADC using Adafruit web application.

5.1 Results via Oscilloscope

The screenshot of oscilloscope, given in Figure 3A, represents the idle state of the baby monitor. In other words, no pressure has been detected. As shown in the image, the observed amplitude is 376mV. The Figure 3B shows the behavior of device in the process of pressure being applied. As observed in the image, the half of the sine waves still have the amplitude of the initial state, while the other half demonstrates the significantly reduced amplitude. After pressing the fabric for more than 5 seconds, the sine waves were stabilized with the amplitude of 160mV. This difference in the amplitudes will be discussed in the Discussion section.

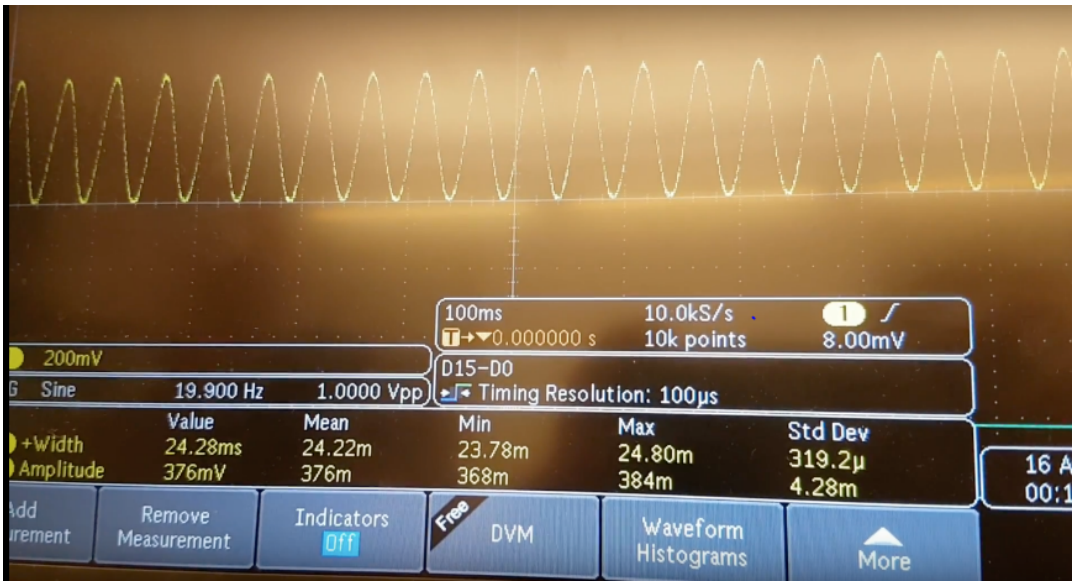

Figure 3.a – View of the Oscilloscope Prior to any applied pressure to the fabric

Figure 3.b – View of the Oscilloscope in the process of applying pressure

Figure 3.c – View of the Oscilloscope after the applied pressure was held for 5 seconds or more

5.2 Results via Web Application

On the other hand, the result of the voltage difference is also recorded via the developed web application. As shown in Figure 4, the baby monitor web application tracks the voltage read via the ADC pin in a graph. If decrease in the voltage value is detected, the baby status alerts the user about its detection.

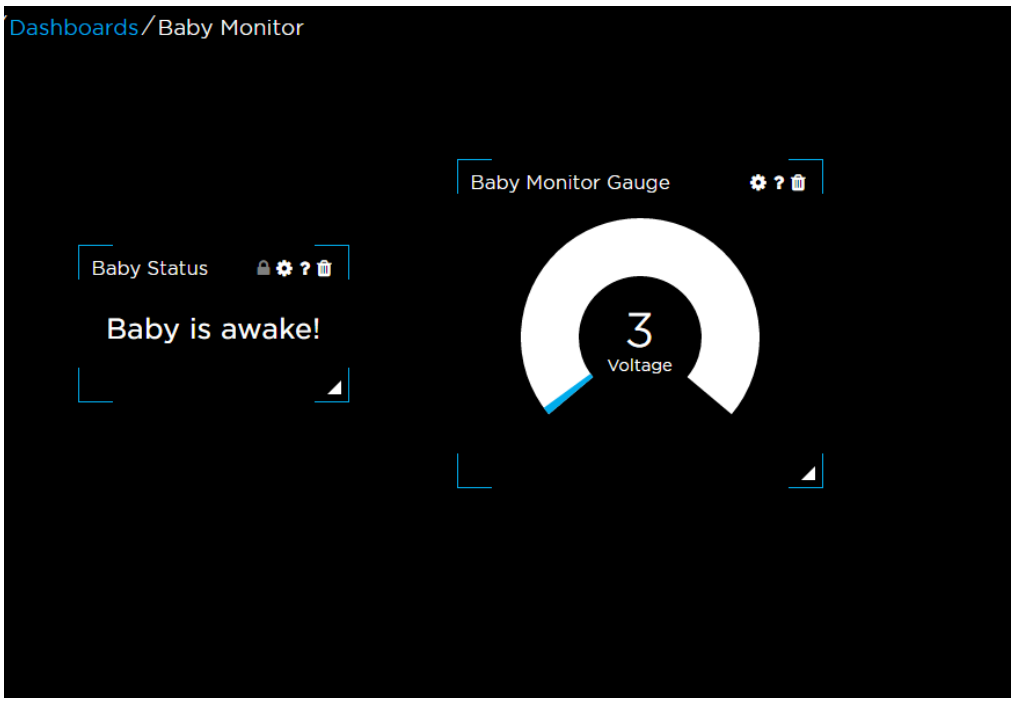

Figure 4 – The screenshot of the web application after the applied pressure was held for 5 seconds or more

6. DISCUSSION

6.1 System Design

Our design used a Tektronix oscilloscope to supply (via arbitrary function generator feature) an AC power source across the Baby monitoring fabric. The power supply was AC rather than DC due to inductive like properties maintained by the fabric where it behaves as a short with a DC signal and drops voltage only when the power supplied is AC. Through trial and error, we determined the optimal frequency to garner the best response to be at 25 Hz. Throughout the experience, the fabric is mounted on 3D printed holder provided to us by Prof. Ebrahim Ghafar-Zadeh's group. We then use another alligator clip and copper wiring to create a conducting path towards the ADC pin of the ESP8266 Huzzah board. The Huzzah board was chosen due to its ten bit resolution at its ADC, built in Wi-Fi capability,

compatibility with the Arduino IDE, serial rate of 115200 baud and a 20 KHz ADC sampling rate. It's 10 bit resolution and 20 KHz ADC sampling rate would be more than sufficient for accurate representing the signal reading taken from the monitoring fabric (by Nyquist theorem, we need a minimum of 50 Hz). It's built in Wi-Fi capability allows us to transmit our data to a remote server. Its fast baud rate means that flashing the board is quick. Its compatibility with the Arduino IDE implies for an increased chance of community forums which we can engage with should be run into any issues. Once readings are taken from the ADC pin, we use the MQTT protocol to communicate with the Adafruit servers. The Adafruit server is where we host our web application that then presents our readings in an aesthetic manner.

6.2 Physics Explanation

The design uses an AC power supply across the fabric since it maintains inductive properties in the sense that it behaves as a short when DC power is supplied. Through trial and error, we determined the best response to be at 25 Hz. A jumper cable is then clipped onto the fabric and using copper wiring and the conductive paths of the bread board, we create a path between the fabric and ADC pin of the Huzzah board. Due to the high impedance at the ADC pin, there should be no current draw to the board (or very minimal). In order to represent our signal, we have to sample at a rate of at least 50 Hz (by Nyquist theorem). The ADC pin samples at a rate of 20 KHz – more than sufficient for what we need. The ADC pin also maintains a 10 bit resolution. This implies that the data taken by our board will provide for an accurate representation of the signal that we read. The ADC pin can take a maximum of 1V – thus, we make sure that the max of the signal driven across the fabric is 1V. Upon receiving data, the board then uses Wi-Fi to

communicate data to the Adafruit servers (where our web application is hosted) via MTTQ communications protocol.

6.3 Software Explanation

For the software part of the project, we used the Arduino IDE for programming in the C language because our board was compatible and for the simple workflow for compilation and flashing that the IDE provides (ensure that proper port is selected and then upload). We use APIs for ESP8266 Wi-Fi capability and for MQTT communication. The setup () function connects to a Wi-Fi hotspot and connects the MTTQ broker to the Adafruit servers (the Adafruit server is like a client). The MQTT protocol is a light weight publish and subscribe communications protocol built on top of the TCP/IP protocol stack. In order to use the protocol, we need to specific clients to the MQTT broker (in this case the Ada Fruit servers). The MQTT broker uses the publish and subscribe design pattern to notify clients per every state change (state change associated to value read by the A0 pin). In the loop () function, we read the analog pin (A0) and the MTTQ broker determines a state change. Web applications designed using the Adafruit servers provide for drag and drop functionality that we can use to create graphs, gauges and any other widget we may need in order to aesthetically present the information that we read.

6.4 Issues Encountered

One of the first issues that we encountered when working with our board was that we would find that if the ADC pin were to read above 1 V, it would shut down (presumably entering reset mode). We would drive an AC voltage of exactly one volt but we found it to often be the case that the system would shut down. When we lowered our AC power to 750 mv peak to peak, the system resumed working as we

expected it to. We attributed the issue to the effects of inherent and unavoidable noise. We also made sure to provide a voltage offset so that our signal would never drive a negative voltage. This is because the ADC of the huzzah board can only accept a voltage range of between [0, 1] V.

Another issue that was encountered was that we were originally working with a Wemos D1 mini (See Figure 7). This board was provided to us by Navid Mohaghegh. Despite initially working, at one point, the board began to fail reading analog inputs - perhaps external damage by us or during manufacturing. When we recognized the issue with the board, we switched to the ESP8266 Huzzah.

Another issue was that we initially were not aware of the inductive properties of the board. We spent some time trying to understand why we not seeing what we expected on the oscilloscopes. This issue was resolved after speaking to Prof. Ebrahim Ghafar-Zadeh.

At one point, we had to switch our fabric holders due to issues with mounts not establishing good enough contacts with the fabric – thus compromising the system (See Figure 5). With the second holder, the mounts were able to establish good contacts with the underlying fabric sensor (See Figure 6).

When we initially established a conducting path between the fabric and ADC pint (via copper wiring) – we took note of the fact there appeared to be a current draw into the ADC pin (despite the fact that there should have been high impedance). To resolve this issue, we used a 100 KOhm resistor to as a limiter on the conducting path (See Figure 8).

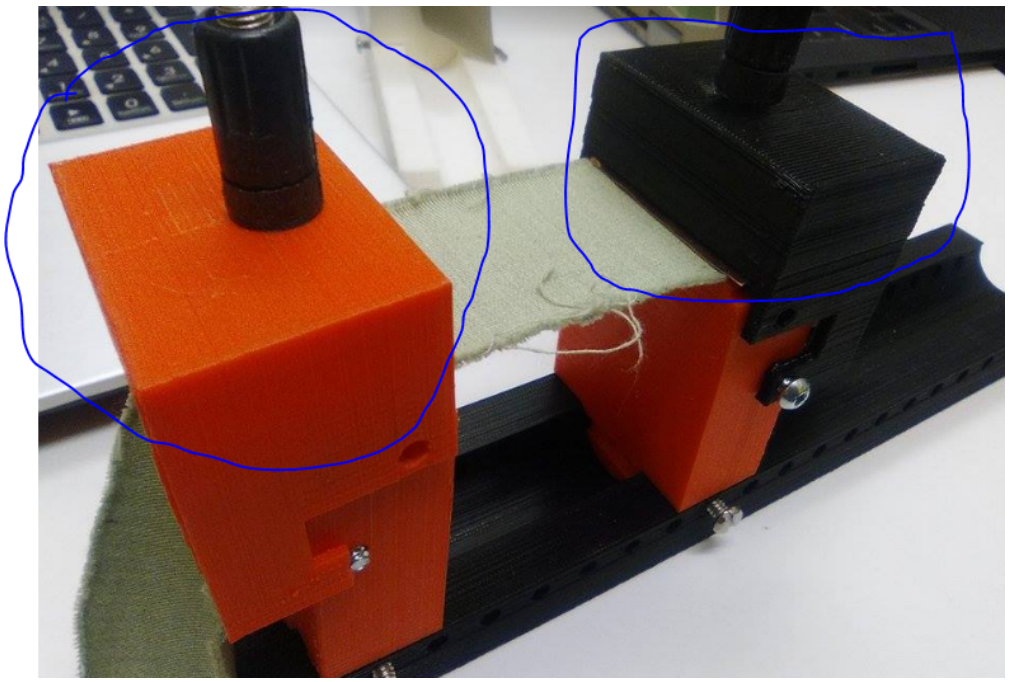

Figure 5 – Fabric Stand 1: *The blue circled sections represent the mounts with which do not enact as strong contact mounts.*

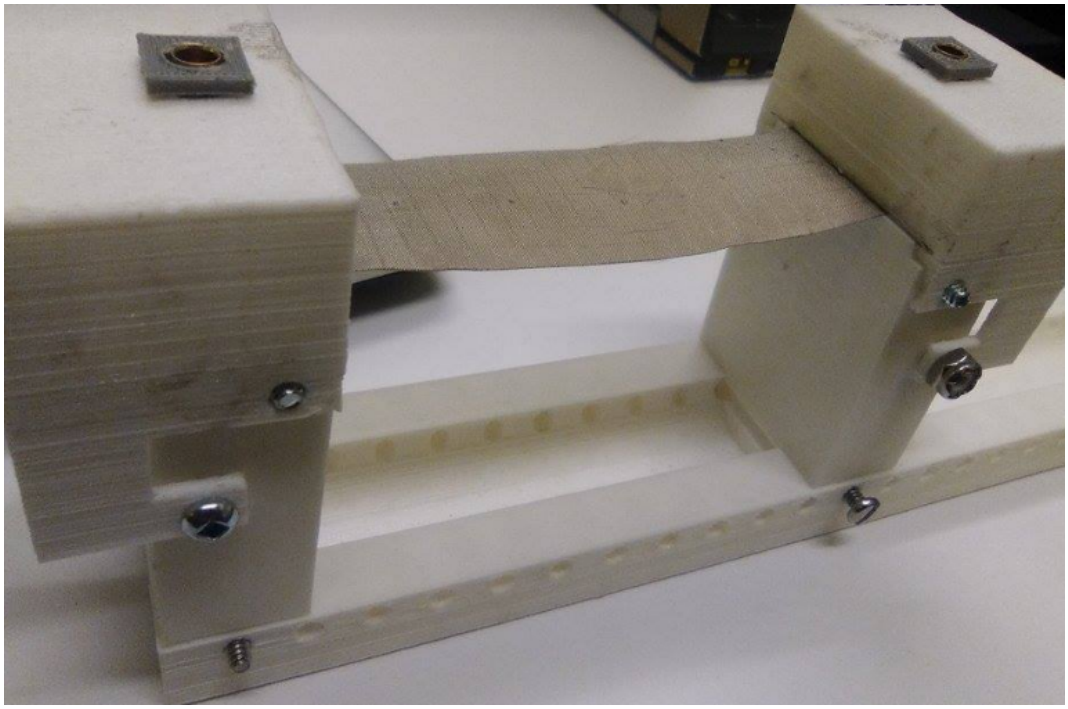

Figure 5 – Fabric Stand 2

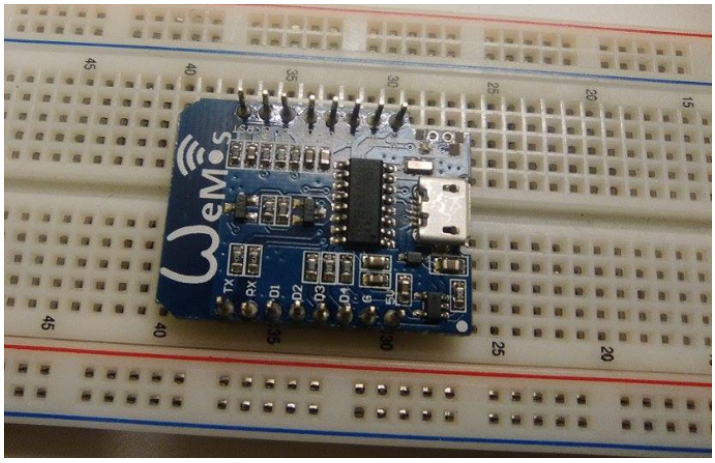

Figure 7 – WeMos mini D1

Figure 8 – 100 KOhm Resistor

7. CONCLUSION

The goal of our project was to demonstrate the knowledge acquired in class and apply that knowledge towards a real world topic. For many of our team members, this was in fact the first time (aside from the labs) to use an embedded system in a project. Despite many issues and setbacks that occurred throughout the development process, we were eventually successful and were able to demonstrate

the characteristics of the baby monitoring fabric. We could not have achieved as much as we did without help from Prof. Ebrahim Ghafar-Zadeh and from research students Samal Munidasa and Parasatoo Baghaei. Their advice was invaluable to our team being able to develop a working system.

In terms of the project, the baby monitor developed in this particular project is a prototype version. This baby monitor can be expanded by placing multiple rows of the fabric strips used in the project or using a larger-sized fabric. By expanding the scale of the project, it is possible to obtain the voltage change from different strips and create a 2D array-based heat-map. This heat-map would not only help the user know the exact location of their baby but also improve the accuracy of the overall project.

Appendix

```
/* Author: Ekram Bhuiyan, Eun Diana Lee, Rishahb Bhat */

/* Includes */
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

/* Definitions */

#define ssid                "BH123"
#define password            "EECBERogers"
/* Required to add web application as a observer */
#define mqtt_server         "io.adafruit.com"
#define mqtt_server_port    1883
#define mqtt_user           "david1196"
#define mqtt_key            "9250bff8a5b84d3480e827115f7589f2"

/* Feed addresses for MQTT Feeds */
#define USERNAME            "david1196/"
#define PREAMBLE            "feeds/"
#define T_GAUGE      "Baby Monitor Sensor"

#define T_CLIENTSTATUS      "clientStatus"
#define T_COMMAND           "command"

#define TRIG  16
#define ECHO  2

/* Global variables */
unsigned long entry;
byte clientStatus, prevClientStatus = 99;

float voltage
char valueStr[5];
WiFiClient WiFiClient;
PubSubClient client(WiFiClient);
```

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(115200);
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(2000);
  Serial.print(".");
  }
  Serial.println("");
  digitalWrite(LED_BUILTIN, LOW );
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  WiFi.printDiag(Serial);
  client.setServer(mqtt_server, mqtt_server_port);
  client.setCallback(callback);
}


void loop() {
  yield();
  if (!client.connected()) {
    Serial.println("Attempting MQTT connection...");
    /* Attempt to add the adadruit servers as an observer */
    if (client.connect("Baby monitor Gauge", mqtt_user, mqtt_key)) {
      Serial.println("connected to MQTT Broker");
      client.subscribe(USERNAME PREAMBLE T_COMMAND, 1);
      client.subscribe(USERNAME PREAMBLE "test", 1);
    }
    else {
      Serial.print("failed, rc=");
```

```
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
  if (client.connected()) {
    for (int j = 1; j <= 10; j++)
    {
        // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float acceleration = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(acceleration);
      Serial.println("Publish voltage");
      String hi = (String)acceleration;
      hi.toCharArray(valueStr, 4);
      client.publish(USERNAME PREAMBLE T_GAUGE, valueStr);
      delay(1000);
    }

  if (client.connected()&& prevClientStatus != clientStatus ) {
    Serial.println("Publish Status");
    String hi = (String)clientStatus;
    hi.toCharArray(valueStr, 5);
    client.publish(USERNAME PREAMBLE T_CLIENTSTATUS, valueStr);
    prevClientStatus = clientStatus;
  }
  client.loop();
}

void callback(char* topic, byte * data, unsigned int length) {
  //  handle message arrived {
  Serial.print(topic);
  Serial.print(": ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)data[i]);
  }
  Serial.println();
```

```
  if (data[1] == 'F')  {
    clientStatus = 0;
  } else {
    clientStatus = 1;
  }
}
```