8 Channel 10-Bit ADC Sensor Reading

Intro to Embedded Systems – EE437
Fall 2023

By

Allen, Peyton T. (allen27@uab.edu)

Diggs, Thomas A. (diggst@uab.edu)

Gavrikov, Yevgeniy K. (yevgeniy@uab.edu)

Instructor: Dr. Abi Yildirim

Abstract:

An analog to digital converter allows one to convert an analog signal into a digital representation. Using a ten 10-bit ADC, data from eight peripheral device sensors can be recorded simultaneously as an array which can then be saved to an SD card, displayed on an integrated LCD screen, and communicated through Bluetooth to a smartphone application for further analysis.

Section A – Project Description

1. Project Description:

This project is dedicated to the Introduction to Embedded System class also known as EE 437 at the University of Alabama at Birmingham. The project given to Team 3 for this semester is to create a sensor monitoring system featuring an 8-channel 10-Bit analog to digital converter (ADC), a 4x20 alphanumeric display, an SD Card data storage utilizing an Excel document, and a wireless connection to an application via Bluetooth low energy. This system will read data from 8 channels from up to 8 sensors which is then converted to digital form and stored for analysis with real time data display on both the alphanumeric display and the application found through a smartphone.

2. Significant System Components

- Liquid-crystal display (LCD)
- Sparkfun (Arduino Uno clone)
- Analog-to-Digital Converter (MCP3008)
- Operation-Amplifier (LM324N)
- Bluetooth module (HC-06)
- Android Smartphone or Tablet
- SD-card Shield
- Potentiometer

3. Constraint Analysis:

There are several limitations and capabilities that have been discovered over the course of working on this project. The limitations have been overcome from research and development throughout the semester. The capabilities were noticed as well while familiarizing ourselves with the hardware and software.

The power efficiency of the device is a limitation to the system. When idling, the device is continuously drawing power. Without interruptions in the software, there is no idle or sleep mode to reduce the power input when data is not being used or received. When writing the code, the refresh rate was set to 10 milliseconds to avoid a refresh rate beyond the capacity of the LCD (~16ms).

In the same vein, the capabilities of the system have not been fully met but can be further realized with the proposed improvements moving forward. The refresh rate capabilities were met by realizing the limitations, yet the other components of this system have not met their limitations in the development of this project.

4. Hardware Interface:

The hardware for our project converts the analog data obtained from a sensor to digital data for a microprocessor to display on an LCD and sends it to a Bluetooth module for displaying the data on a smartphone/tablet application. The analog data is input by a sensor, which is a potentiometer in our test cases. The analog data is sent through an op-amp to prevent a voltage difference between the input and output pins and to control the voltage being sent into the analog-to-digital converter (ADC).

The ADC receives analog data and converts to digital for the microprocessor (we used a Sparkfun which is a clone of the Arduino Uno Rev3) to use. The ADC chip select is placed on pin 3 to allow for the SD card to use the Arduino's chip select pin. The microprocessor outputs the digital data to three locations: an LCD, Bluetooth module, and SD card reader. A 16-pin LCD is attached to a 4-pin I2C adapter to transmit data to Arduino's I2C pins (SDA and SCL). A single 100k Ohm resistor is grounding each input into the ADC to prevent floating numbers.

To send the data to a SD card, a SD card shield is placed over the Sparkfun which connects an SD card reader to the MOSI, MISO, CLK, and CS pins of the Arduino (pin 10-13). The HC-06 Bluetooth module uses 4 pins: a 5V source, GND, RX, and TX. The TX (transmitting) and RX (receiving) pins on the Bluetooth module are coded for and connected to pins 6 and 7 respectively to allow for continuous uploading to the module. The addition of the SD card shield was easily integrated into the existing circuit. After the wires were unplugged from the Sparkfun, the SD-card shield was aligned with the sockets for the Sparkfun, and the wires were reconnected into the SD-card shield sockets.

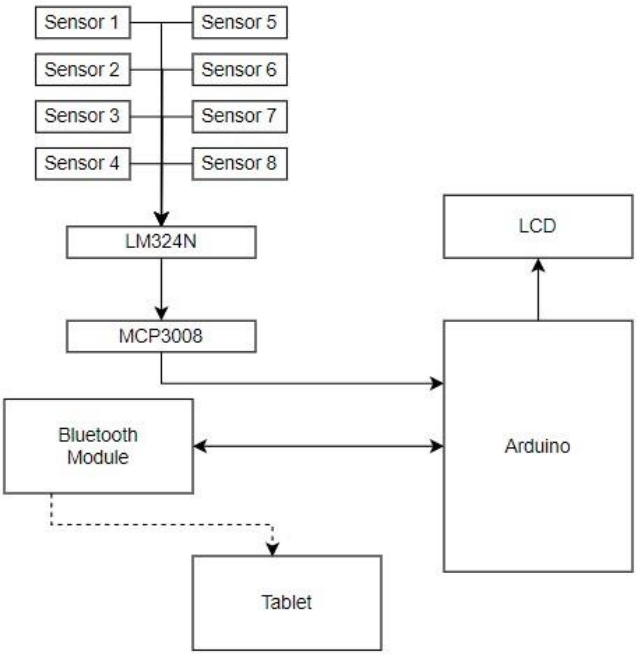

Figure 1: Hardware Block Diagram

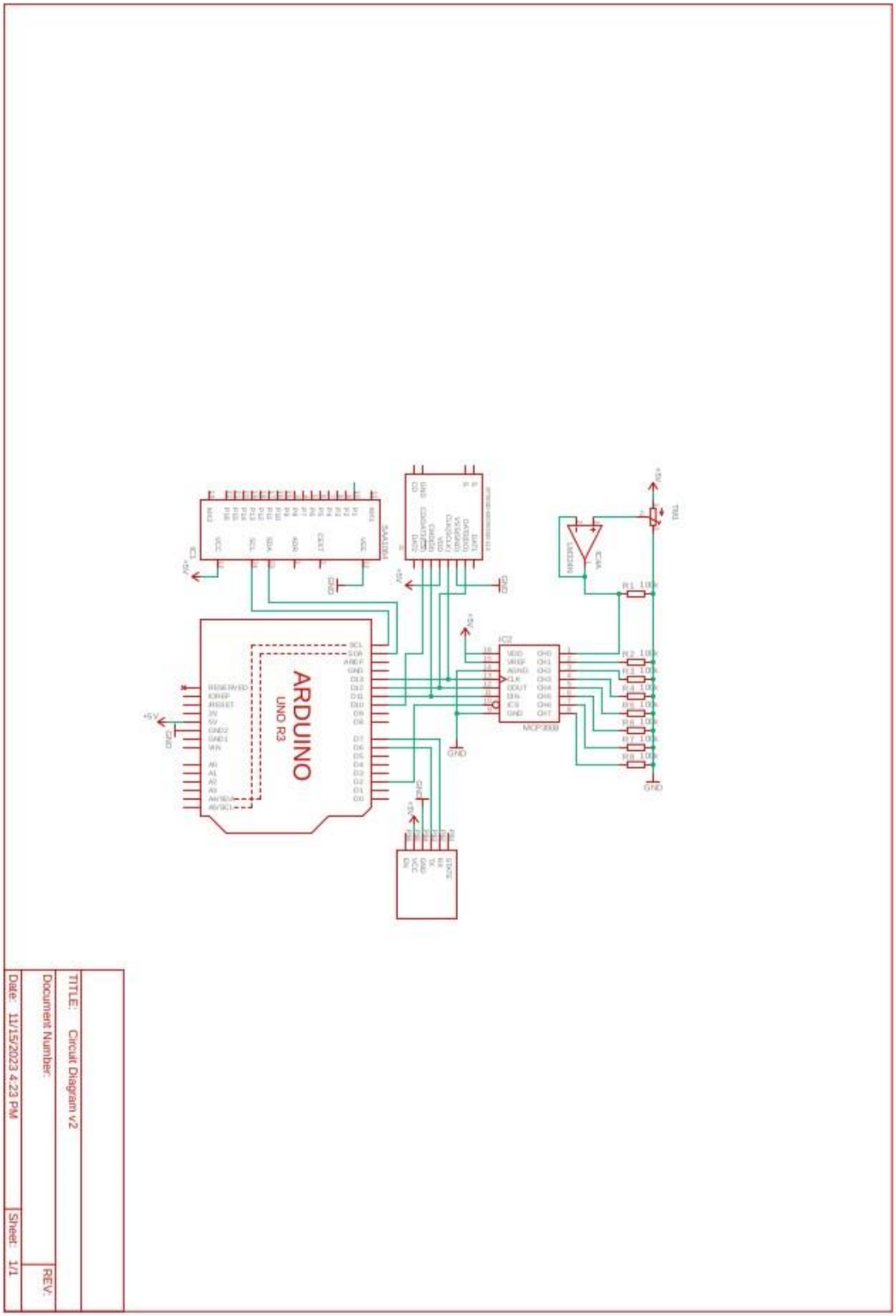

Figure 2: Hardware Circuit Schematic through Eagle

5. Software Interface:

Arduino:

The software interface is programmed in Arduino IDE programming language which is like C++ and C# languages. This program starts with initializing the ADC, the alphanumeric display, the SD card, and the BLE. The sensor data is then transferred through the operation amplifier and the analog to digital converter to the Arduino to be read. The ADC steps are read in and then converted to millivolts by dividing the steps received by total steps (1024) and multiplying the 5000 mV. The data produced from every sensor is sent through Bluetooth. Data is relayed through serial communication in hexadecimal code to help with package size.

The LCD is cleared before anything is sent to it, to ensure that a blank space is used. Units and formatting specifications are sent to the LCD screen, such as C1-8, delimiter | borders, and mV units. This allows the data to be read clearly. The data for each sensor is sent directly to the respective channel positions, creating a full table of sensor channel data.

The SD Card is formatted in the program using a file write function. The data of each channel is sent to the SD Card, creating a .csv file with the data. This SD Card can be removed at any time and placed into a laptop, allowing the user to have an excel folder of the data present.

This data is sent in five samples per second. Therefore, the sensor data is updated every 200 milliseconds. The LCD is cleared every 200 milliseconds as well, this creates a display that is constantly updated in close to real time. Bluetooth serial data is sent at the same rate with a 9600 or varying as needed baud rate. The SD card data is sent at a slower rate to limit the values sent to the excel sheet.

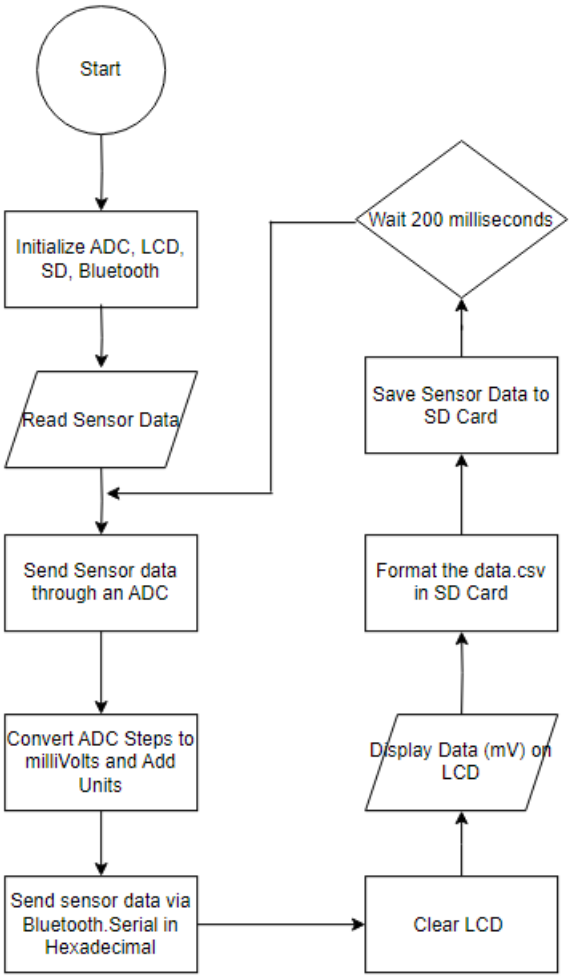

Figure 3: Arduino Software Flow chart

Tablet Software and Interface:

The tablet software was created for the Android operating system using the online resource, MIT App Inventor 2 (MAI2). The software is accessible either as a standalone application as a .apk file as though downloaded from Google Play app store, or it is accessible by downloading the MAI2 application and using the project's authenticator-generated code. This latter method allows for Android and IOS accessibility, and it allows for real time editing of the application even when connected to the HC-06 Bluetooth low energy (BLE) module. The service provided by MAI2 allows one to create an application using the online portal. This web service allows one to edit the user interface as shown in Figure 5 and Figure 6. MAI2 also allows one to edit the underlying code for the application with a convenient block visualization as shown in Figure 7 and in the appendix and accompanying Prezi presentation. This block representation mirrors an integrated development environment yet has documentation-based draw backs that standard IDEs do not have due to the necessity for custom extensions such as those utilized for the Bluetooth low energy components; MAI2 natively supports regular Bluetooth but not the low energy counterpart, so one needs to find user uploaded extensions instead of relying on open sourced libraries such as when coding with Python or Java.

The data recorded by the Arduino software is saved to an SD card, displayed on the LCD screen, and then, the serial data is transmitted through the BLE module and displayed in the application. The block diagram flowchart of the application is shown in Figure 4. For the application to connect with the Arduino, one must enable Bluetooth and Location Services on a device with Android software version 10 or newer. After opening the application, one can connect to the 10 Bit 8 Channel ADC by scanning and selecting the device found under the name "EE437_Team3." Once connected, the application displays a successful connection message along with the Arduino Bluetooth module's hardware information. The user can then select one of the 8 channels which corresponds to the connected accompanying peripheral device such as the potentiometer on channel 1. From there, the user can monitor the data in real time in graph format, pause to step through the buffered chart data, or reset the graph. Also at any point, one can select a different channel, minimize or maximize details about received data, disconnect from the Arduino Bluetooth module to select a different project, or one can exit from the application entirely.

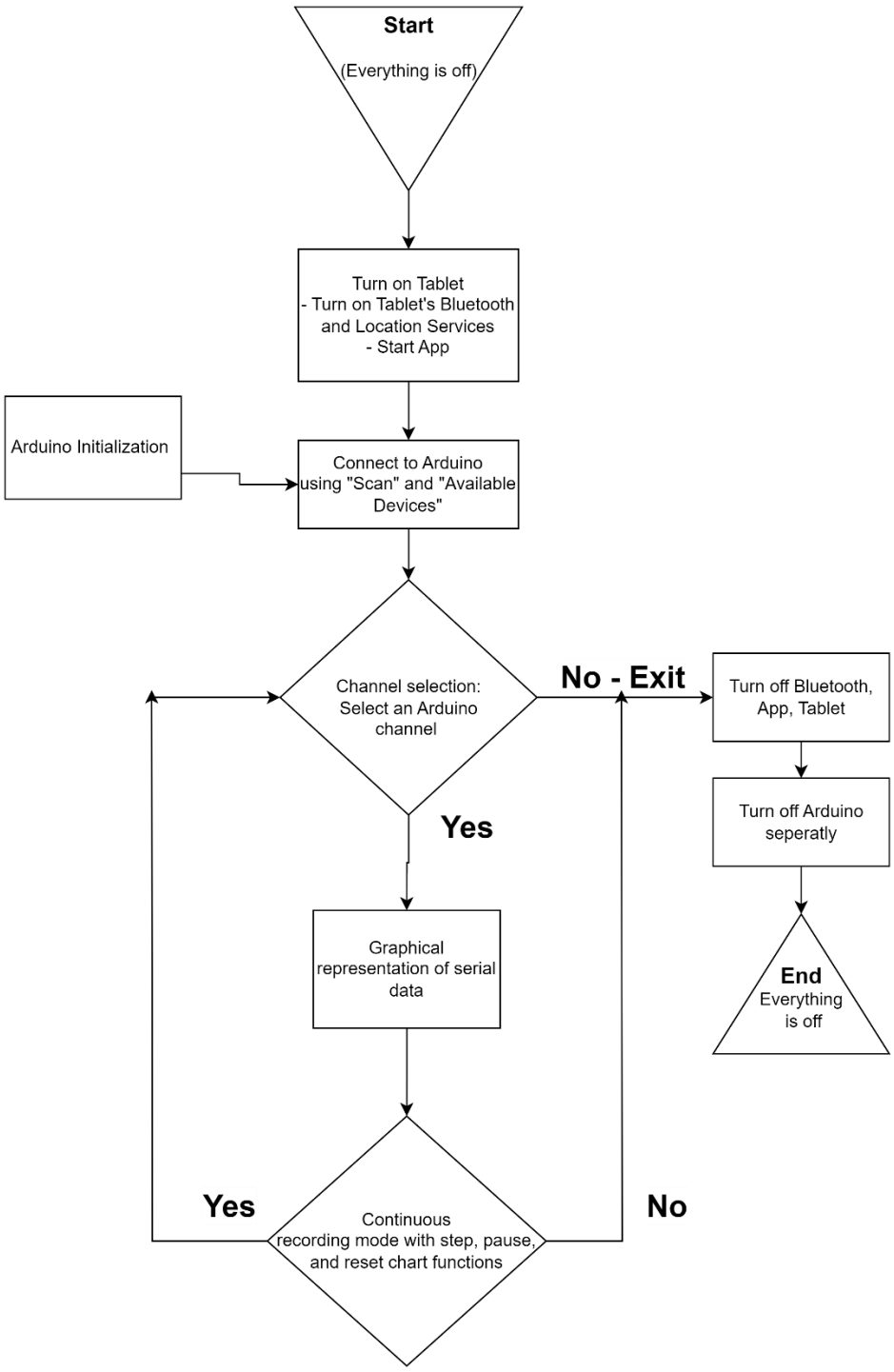

Figure 4: Tablet Software Block Diagram

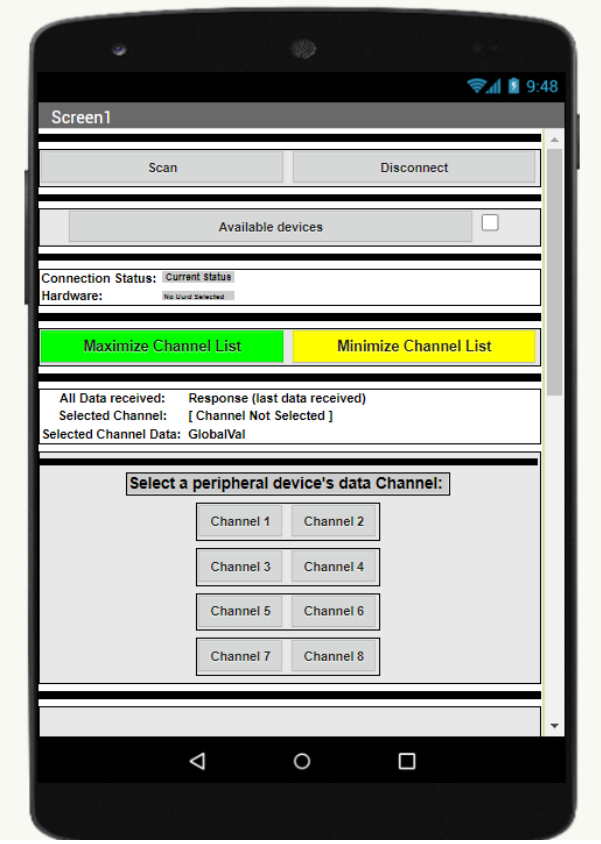

Figure 5: Tablet Software - Application User Interface 1 (GUI)

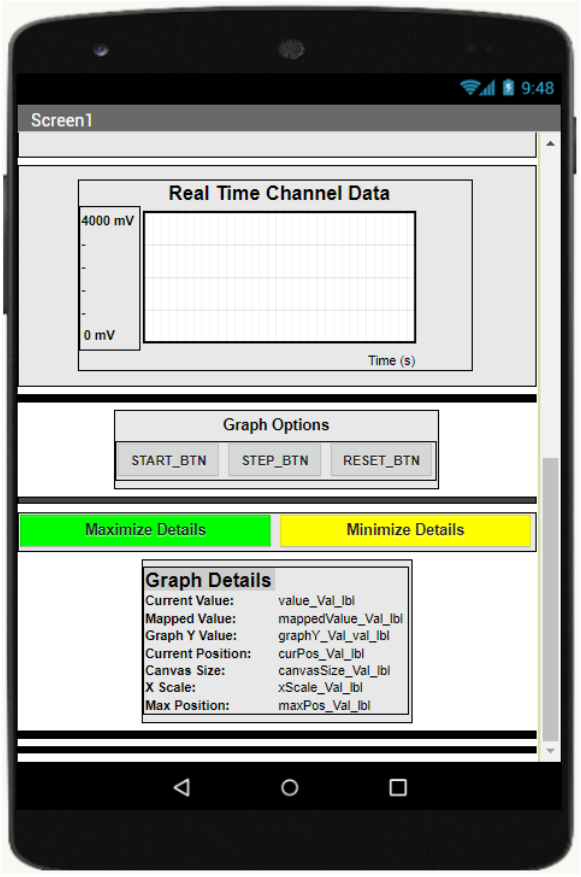

Figure 6: Tablet Software: Application User Interface 2 (GUI)

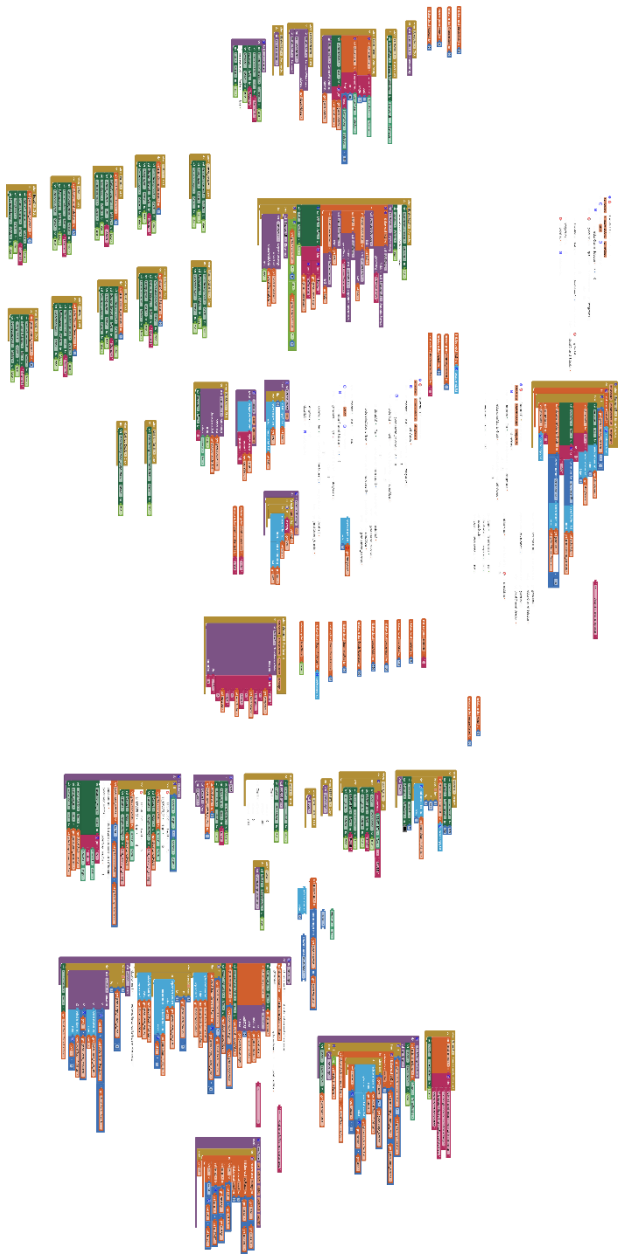

Figure 7: Tablet Software - Application MIT App Inventor Block Diagram

6. Power Constraints and Efficiency:

For this project, a few power constraints were met during the construction of the circuitry. The overall power draw of the circuit could be improved by using interruptions to stop or delay the actions of the hardware. As it stands, the device is drawing continuously, and with this change, it can be modified for further improvements including portability.

7. Cost Constraints:

The main cost constraints relate to the supplied kit components. Team 3 did not damage any components during the design and prototyping phase which eliminated any additional costs as all project components can be reused for future projects. The cost of the Android tablet or licensing is not included as the application is available for public usage through MAI2 on most smartphone devices.

8. Component Selection Rationale:

	Component #1	**Component #2**	**Component #3**
Microprocessor	Sparkfun REDBOARD	Arduino ATMega328	Raspberry Pi SC0563
LCD	Newhaven Display NHD-0420DZ	SunFounder I2C – CN0296D	Orient Display – AMC2004AR
Analog-to-Digital Converter	Adafruit – MCP3008	Sparkfun PID 13906 Multiplexer	Texas Instruments – TLC0838IN
Operation-Amplifier	Texas Instruments LM358P	Texas Instruments LM324N	Texas Instruments TL9721P
Bluetooth module	Bluetooth Module HC-05	Bluetooth Module HC-10	Bluetooth Module HC-06
SD-card reader	ESP8266 MicroSD reader Module	HiLetgo Shield w/ SD Card reader	Seeed Studio SD card V4.0 Shield

The components used in this project were provided by Dr. Abi throughout the semester.

Section B – Project Details

Significant System Components:

- Liquid-crystal display (LCD): the liquid-crystal display is used as a digital display for the voltage values of each individual channel in an orderly and customizable format; the refresh rate of every 10 milliseconds allows for a regular feed of data to the display screen.
- Sparkfun (Arduino Uno clone): the Sparkfun is the microprocessor used for this device which we have programmed to communicate between all external and internal hardware components.
- Analog-to-Digital Converter (MCP3008): a Texas Instruments integrated chip was used to provide eight channel inputs for the analog to digital conversion; the digital data is then input into the Arduino digital input ports.
- Operation-Amplifier (LM324N): a Texas Instruments integrated chip is used for the operation amplifier; this chip amplifies the analog data from the sensors to the ADC and prevents voltage feedback from damaging the ADC.
- Bluetooth module (HC-06): used for serial data communication between the Arduino and tablet.
- Android Smartphone or Tablet: Android version 10 or higher is required as newer software versions support Bluetooth low energy mode.
- SD-card Shield: the SD-card shield is used to insert a microSD and microSD-to-SD adapter and read data on to the microSD; the SD card shield is aligned and placed directly into the sockets of the Sparkfun.
- Potentiometer: the potentiometer is the sensor that we chose to give us an analog data input for testing.

Detailed Project Description

Part 1 – Arduino and Breadboard setup:

Step 1: Arduino setup: install the Arduino board on the breadboard.

Step 2: LCD screen setup: install on breadboard and connect to corresponding Arduino pins.

Step 3: ADC component setup: install on breadboard and connect to corresponding Arduino pins.

Step 4: Op-Amp component setup: install on breadboard and connect to sensor and ADC.

Step 5: HC-06 Bluetooth low energy module: install the module on the breadboard. Connect the BLE rx and tx output with the corresponding pins as detailed in the Arduino code for the corresponding tx and rx inputs. (pins 7 and 8). Use the Arduino's ground and 5volt breadboard row for power requirements.

Step 6: Wi-Fi shield component setup: install on the Arduino board and connect to corresponding Arduino pins.

The resulting assembly is shown in Figure 8.

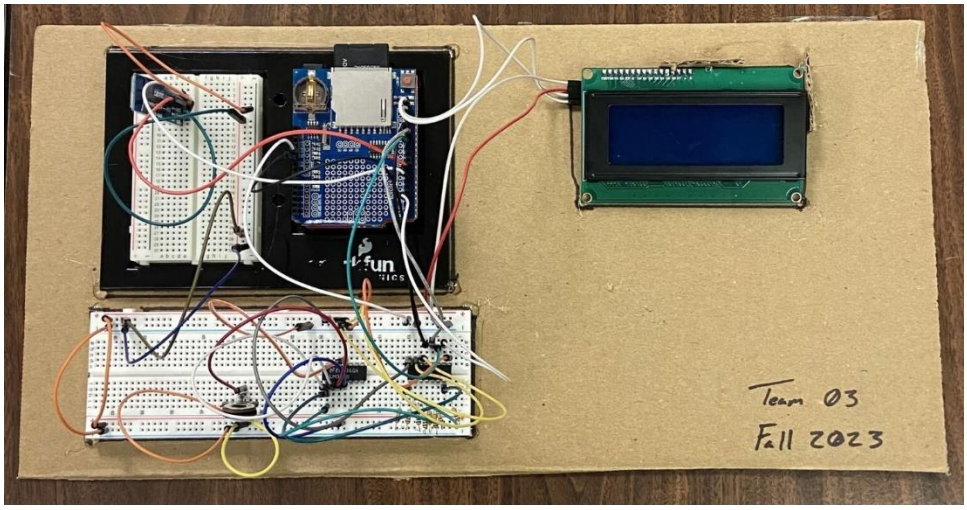

Figure 8: Completed 8 Channel 10-Bit ADC Sensor Assembly

Part 2 - Creating the Tablet application:

Step 7: Register an account at the MIT App Inventor 2 website.

Step 8: Research documentation for the BLE extension and Jaun's canvas blocks.

Step 9: For the BLE block section, create a scan and a connect button that captures the hardware information of the HC-06 module.

Step 10: For the received serial data string, create an indexing section that corresponds to the 8 channel select buttons.

Step 11: Once a channel is selected, feed that data into the canvas block section that iteratively creates a slide left graph using the tutorials provided by Juan Antonio. Using the backpack feature in the MIT portal, one can copy and paste relevant code, yet troubleshooting requires trial and error in both the UI and block code sections as either one can create interface, memory, or buffering errors.

Results

This project resulted in many successes. One success was the LCD screen displaying all the channel's sensor data properly. As one can see in Figure 9, the channel number, the data pulled from the Arduino, the unit of milli volts, and a border to make a distinction between the channels.

Figure 9: LCD Results

Similarly, success was met with the Android application's data capturing and graphing as shown in Figure 10. The figure shows all data received which is all eight of the channel data. The app then determines the selected channel and displays the channel data selected. The graph below shows the changes made to the sensor and continues data when switching to other channels.

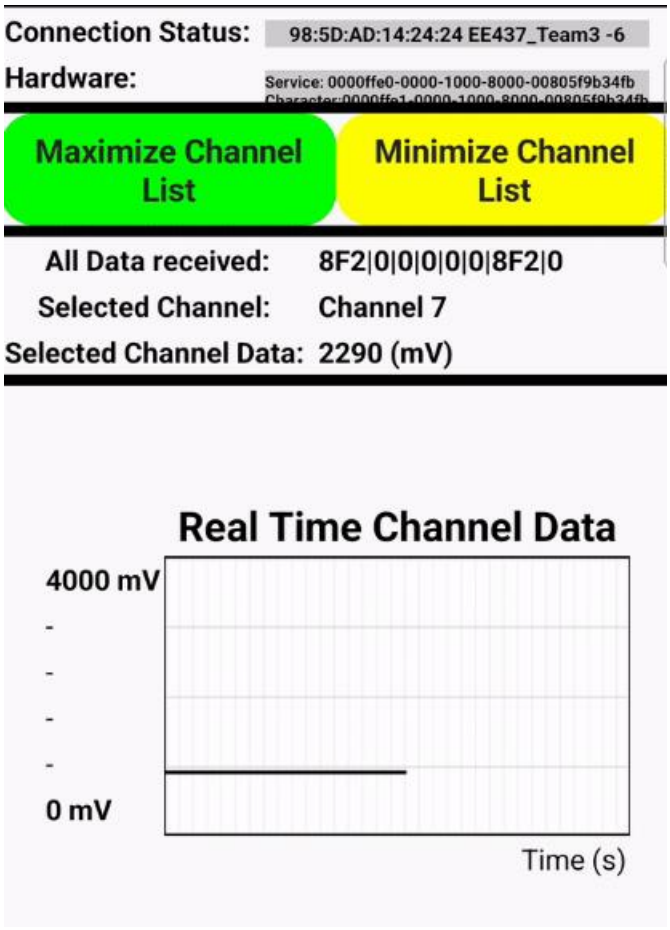

Figure 10: MIT App Inventor Results

Future Improvements

For the Arduino, wire management and housing would be improvements to the aesthetic design and simplicity of the breadboard. For the tablet application, improvements can be made to the aesthetics of the graphical interface and to the graphing section to include an export of the displayed data. With the hardware, the addition of an external rechargeable battery could improve the portability of the device.

Team organization and labor distribution

a. **Peyton Allen:** oversaw the hardware aspects of the project. His duties included properly wiring the sensor, ADC, amplifier IC, and LCD to the Arduino. He is also responsible for the block diagrams and any CAD/Eagle software schematics for the circuitry.

b. **Thomas Diggs:** oversaw writing and formatting the software for the Arduino and associated hardware with the Arduino. This included writing the code to communicate between the analog-to-digital converter (ADC) integrated chip (IC), the amplifier IC, sensors, and the Arduino; creating a simple user interface for the LCD to display the sensor data; and ensuring the Arduino wrote to a SD card. In addition, he created the software flow chart for the Arduino and has been the float member of the group, helping whoever needs it and building the system.

c. **Yevgeniy Gavrikov:** oversaw integrating the system to a device application. This allowed the Arduino to wirelessly communicate with a wireless device, in this case a smartphone, using Bluetooth. This included learning about and using MIT App Inventor along with a knowledge of AT Commands and other Bluetooth methods.

Section C – Appendix

Arduino Libraries:

1. Adafruit_MCP3008.h (Link Here)
2. LiquidCrystal_I2C.h (Link Here)
3. SoftwareSerial.h (Link Here)
4. SD.h (Link Here)
5. SPI.h (Link Here)

MIT App Inventor Library:

6. MIT App Inventor 2 - BLE extension (Link Here)

Eagle Libraries:

7. abi-lcd
8. arduino_jw
9. hc06
10. mcp3304
11. potentiometers
12. i2c-led-controller

References:

1. LM324-N Op-Amp Integrated Chip datasheet (Link Here)
2. MCP3008 ADC Integrated Chip datasheet: (Link Here)
3. Tablet software - Bluetooth Low Energy extension: (Link Here)
4. Tablet software – Juan Antonio's shift left graph for graphing using the canvas block: (Link Here)
5. Tablet Software – MIT Block Code Presentation: (Link Here)

Arduino Code:

```
// Thomas Diggs, Peyton Allen, Yevgeniy Gavrikov
// UAB - Fall 2023
// EE437

// Include Necessary Libraries
#include <Adafruit_MCP3008.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include "SD.h"
#include "SPI.h"

// Define Constants
#define i2C_LCD_ADDR 0x3F  // I2C address of the LCD
#define NEW_CS_PIN 2       // Chip Select pin for MCP3008
#define TX 6               // Transmit pin for SofwareSerial
#define RX 7               // Receive pin for SoftwareSerial

// Create a SoftwareSerial object for Bluetooth communication
SoftwareSerial BTSerial(RX, TX);

// Create a LiquidCrystal_I2C object for the LCD
LiquidCrystal_I2C lcd(i2C_LCD_ADDR, 20, 4);

// Define Variables
int dly = 500;
int channel1 = 0;
int channel2 = 1;
int channel3 = 2;
int channel4 = 3;
int channel5 = 4;
int channel6 = 5;
int channel7 = 6;
int channel8 = 7;

// Create an Adafruit_MCP3008 object for the MCP3008 ADC
Adafruit_MCP3008 mcp;

// Define variables for SD Card
const int CSpin = 10;    // Chip Select pin for the SD Card
String dataString = "";  // Holds the data to be written to the SD card
int voltval1 = 0;        // Value read from your first sensor
int voltval2 = 0;        // Value read from your second sensor
File sensorData;         // File object for SD Card
```

```
int count = 0;           // Counter for data recording

// Setup function
void setup() {
  BTSerial.begin(9600);  // Start SoftwareSerial for Bluetooth communication
  Serial.begin(9600);    // Start serial communication with the computer
  Serial.println("Maker's Digest: Ready");
  Serial.print("Initializing SD card...");

  mcp.begin(NEW_CS_PIN);  // Initialize MCP3008 ADC with the specified Chip
Select pin

  lcd.init();  // Initialize the LCD
  lcd.backlight();

  // SD Card - see if the card is present and can be initialized:
  if (!SD.begin(CSpin)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    return;
  }
  Serial.println("card initialized.");
}

// Loop function
void loop() {
  char msg[50];

  // Read sensor values from MCP3008 ADC
  int val1 = mcp.readADC(channel1);
  int val2 = mcp.readADC(channel2);
  int val3 = mcp.readADC(channel3);
  int val4 = mcp.readADC(channel4);
  int val5 = mcp.readADC(channel5);
  int val6 = mcp.readADC(channel6);
  int val7 = mcp.readADC(channel7);
  int val8 = mcp.readADC(channel8);

  // Convert ADC values to voltages
  int voltval1 = (val1 / 1024.0) * 5000.0;
  int voltval2 = (val2 / 1024.0) * 5000.0;
  int voltval3 = (val3 / 1024.0) * 5000.0;
  int voltval4 = (val4 / 1024.0) * 5000.0;
  int voltval5 = (val5 / 1024.0) * 5000.0;
  int voltval6 = (val6 / 1024.0) * 5000.0;
```

```
int voltval7 = (val7 / 1024.0) * 5000.0;
int voltval8 = (val8 / 1024.0) * 5000.0;

// Clamp voltage values to 0 if below 100 mV
if (voltval1 > 100) {
  voltval1 = voltval1;
} else {
  voltval1 = 0;
}
if (voltval2 > 100) {
  voltval2 = voltval2;
} else {
  voltval2 = 0;
}
if (voltval3 > 100) {
  voltval3 = voltval3;
} else {
  voltval3 = 0;
}
if (voltval4 > 100) {
  voltval4 = voltval4;
} else {
  voltval4 = 0;
}
if (voltval5 > 100) {
  voltval5 = voltval5;
} else {
  voltval5 = 0;
}
if (voltval6 > 100) {
  voltval6 = voltval6;
} else {
  voltval6 = 0;
}
if (voltval7 > 100) {
  voltval7 = voltval7;
} else {
  voltval7 = 0;
}
if (voltval8 > 100) {
  voltval8 = voltval8;
} else {
  voltval8 = 0;
}
```

```
// Bluetooth communication - Send voltage values in hexadecimal
//    this is so the data sent in below 20 characters
BTSerial.print(voltval1, HEX);
BTSerial.print("|");
BTSerial.print(voltval2, HEX);
BTSerial.print("|");
BTSerial.print(voltval3, HEX);
BTSerial.print("|");
BTSerial.print(voltval4, HEX);
BTSerial.print("|");
BTSerial.print(voltval5, HEX);
BTSerial.print("|");
BTSerial.print(voltval6, HEX);
BTSerial.print("|");
BTSerial.print(voltval7, HEX);
BTSerial.print("|");
BTSerial.print(voltval8, HEX);

// LCD update
lcd.clear();

// LCD Display
lcd.setCursor(0, 0);
lcd.print("C1: ");
lcd.setCursor(3, 0);
lcd.print(voltval1);

lcd.setCursor(0, 1);
lcd.print("C2: ");
lcd.setCursor(3, 1);
lcd.print(voltval2);

lcd.setCursor(0, 2);
lcd.print("C3: ");
lcd.setCursor(3, 2);
lcd.print(voltval3);

lcd.setCursor(0, 3);
lcd.print("C4: ");
lcd.setCursor(3, 3);
lcd.print(voltval4);

lcd.setCursor(10, 0);
lcd.print("C5: ");
lcd.setCursor(13, 0);
```

```
  lcd.print(voltval5);

  lcd.setCursor(10, 1);
  lcd.print("C6: ");
  lcd.setCursor(13, 1);
  lcd.print(voltval6);

  lcd.setCursor(10, 2);
  lcd.print("C7: ");
  lcd.setCursor(13, 2);
  lcd.print(voltval7);

  lcd.setCursor(10, 3);
  lcd.print("C8: ");
  lcd.setCursor(13, 3);
  lcd.print(voltval8);

  // Repeat the LCD display setup for other channels (C2 to C8)

  // Additional LCD Formatting
  for (int i = 0; i < 4; i++) {
    lcd.setCursor(7, i);
    lcd.print("mV");

    lcd.setCursor(9, i);
    lcd.print("|");

    lcd.setCursor(18, i);
    lcd.print("mV");
  }

  delay(dly);

  // SD Card - Save data every 10 loops
  count = count + 1;

  if (count >= 10) {
    dataString = String(voltval1) + "," + String(voltval2) + "," +
String(voltval3) + "," + String(voltval4) + "," + String(voltval5) + "," +
String(voltval6) + "," + String(voltval7) + "," + String(voltval8);  //
convert to CSV
    saveData();

                                                        // save to SD card

    delay(100);
```

```
    count = 0;
  }
}

// Function to save data to the SD card
void saveData() {
  // Open the data.csv file for writing
  sensorData = SD.open("data.csv", FILE_WRITE);

  // Check if the file is successfully opened
  if (sensorData) {
    // Check if the file is empty (new file)
    if (sensorData.size() == 0) {
      // If the file is empty, write the header line
      sensorData.println("Channel 1 (mV), Channel 2 (mV), Channel 3 (mV),
Channel 4 (mV), Channel 5 (mV), Channel 6 (mV), Channel 7 (mV), Channel 8
(mV)");
    }
    // for (int i = 0; i < 25; ++i) {
    //   array[i];
    // }

    // Write the dataString to the file
    sensorData.println(dataString);

    // Close the file
    sensorData.close();
  } else {
    // If the file couldn't be opened, print an error message
    Serial.println("Error opening data.csv for writing!");

    // Attempt to create the file
    sensorData = SD.open("data.csv", FILE_WRITE);

    // Check if the file is successfully created
    if (sensorData) {
      // If the file is open, write the header line
      sensorData.println("Channel 1 (mV), Channel 2 (mV), Channel 3 (mV),
Channel 4 (mV), Channel 5 (mV), Channel 6 (mV), Channel 7 (mV), Channel 8
(mV)");

      // Write the dataString to the file
      sensorData.println(dataString);

      // Close the file
```

```
      sensorData.close();

      Serial.println("data.csv created!");
    } else {
      // If the file still couldn't be created, print an error message
      Serial.println("Error creating data.csv!");
    }
  }
}
```