Design Review: Report

Section 01

Dec. 08, 2017

Team CIVIC:

Dongjin Kim

Junyan Zhai

Haoyu Zhang

Veronica Vera

State of Project

Unresolved Issues

- Could not program the microcontroller module on the PCB
- Heart rate reading is sometimes not accurate
- Sometimes only the first reading of the temperature sensor is recorded
- Step counter gets more accurate with increase data gathering
- Components were not placed on PCB
- Rechargeable battery charging not functional

State of Deliverables and Demonstration Test Status

The team was able to demonstrate that the following subsystems worked correctly and interacted with each other:

- Microcontroller / BLE
- Accelerometer
- Thermometer
- Heart rate monitor
- GPS module
- User app (Android)

The only subsystem that stood by itself during the demonstration was the power distribution subsystem, with the 5V and the 3.3V regulation functioning, but not the charging of the rechargeable battery.

Acceptance Test Status

The following acceptance tests passed within our subsystems as stated on the FATs of the team agreement:

- GPS module: GPS reading was accurate within 5 meters.
- BLE: The connection was stable with BLE for an extended amount of time.
- Step counter: The counter was accurate within 5 steps.
- Thermometer: the thermometer was accurate with 3 degrees Celsius.

The following acceptance tests were not met as stated one the FATs of the team agreement:

- Battery measurement: No LEDs showed the charging / discharging states of the battery, or the BLE connection.
- Heart rate monitor: Sensor reading was not accurate within 3 beats per minute

Although the battery and the heart rate monitor subsystems did not meet the acceptance tests of the team agreement, both of these subsystems were still able to gather data.

Table of Contents

User Stories

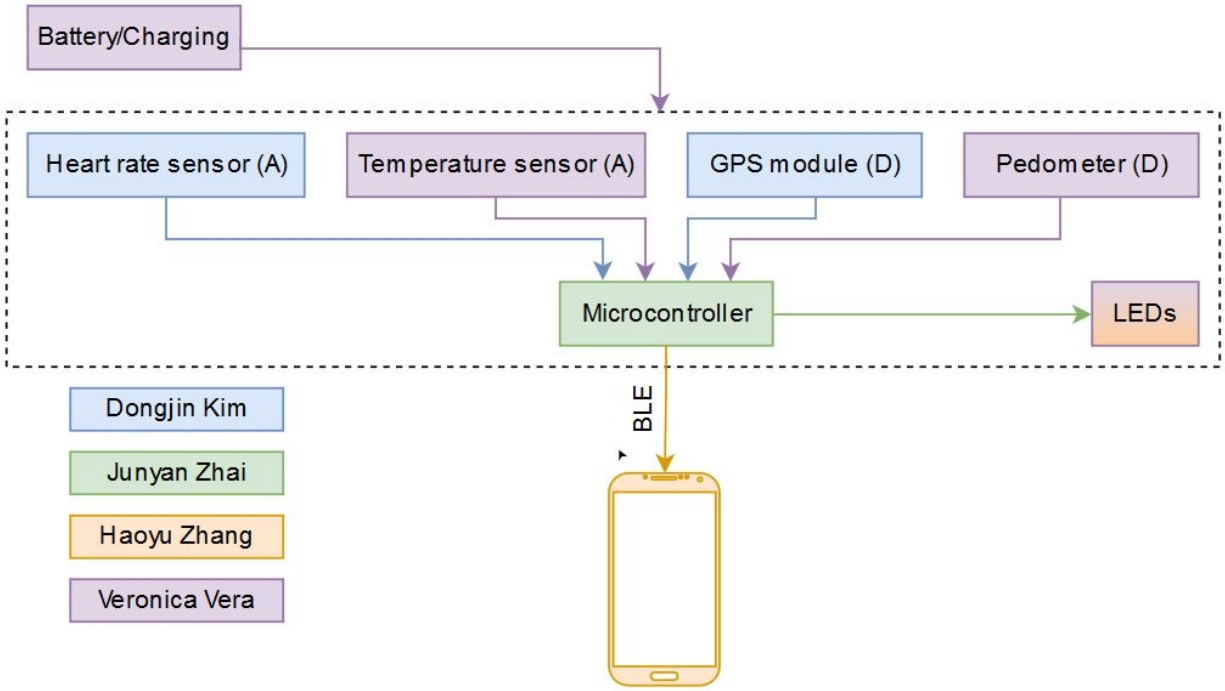
Jane the Jogger

Jane the jogger likes to run in the evenings. Jane has already lost her phone several times during her evening runs but wants to track her workouts without worrying about losing her phone again. Jane does not enjoy running with her phone, because she is clumsy and it might shatter easily if she dropped it while running. While she runs, she also dislikes carrying her phone's weight in her pocket or her hand. However, she likes the idea of knowing tracking her heart rate, temperature and her speed with a single, portable device that has a user-friendly app.

Graham the Grandpa

Graham the grandpa is too old to take care of himself. His family is too busy to check on him constantly, but they worry about his health. Graham has a history of getting high fevers and anxiety, causing his heart rate to raise. Graham is also a very lazy grandpa, but the doctors say he needs to walk regularly to reduce his anxiety and lower his pulse. Graham's family is looking for a device that can warn them in case his temperature or heart rate increase too much, and check if Graham is getting his daily steps in and check in on his location.

System Block Diagram



Setup Instructions

In order to setup the components of the arm band, the user needs to make sure that the battery is connected to the armband, and that the Android device is connected through BLE. Aside from these, with the working prototype, the user needs to make sure all components are connected to 3.3V (aside from the operational amplifier with needs 5V). Lastly, each component should be correctly connected with the following protocols to the microcontroller:

- Heart rate monitor: ADC
- GPS: UART
- Thermometer: ADC
- Accelerometer: SPI

Operating Instructions

Overall product description

Our team will create an armband with a GPS module, heart rate sensor, step counter, thermometer and Bluetooth low energy (BLE) module. The rechargeable battery pack will be charged through a USB power source (5V) and will interact with the user through an Android app. Data will be stored in the armband until downloaded to a phone through BLE and measured in 6-hour intervals. The device will gather step-counting data constantly (not only during intervals). Additionally, the armband starts gathering data upon request from the user. The app will display daily heart rate, step count, temperature, and location (on a map) data. A small red LED will tell the user that the battery is under 20%, and a green LED will indicate that the battery is over 20%. A blue LED will light up while there is a connection to a device through Bluetooth. Lastly, the user can set thresholds for warnings in case the heart rate, temperature or daily step counts reach the thresholds.

MCU and BLE module

In the prototype we designed, we use CY8C4247LQI-BL483 PSoC 4 BLE module. It has a embedded 32-bit microcontroller with 128 KB flash memory. It contains 12-bit, 1-Msps SAR ADC with differential and single-ended modes that can be used for processing the signal from our temperature and heart rate sensor. It also has two independent runtime reconfigurable serial communication blocks (SCBs) with reconfigurable I2C, SPI, or UART functionality that can be used for our pedometer and GPS. It has up to 36 programmable GPIOs that can be used for external input or output such as sensors and LEDs. PSoC Creator provides the development environment for our BLE module.

Schematic of BLE module

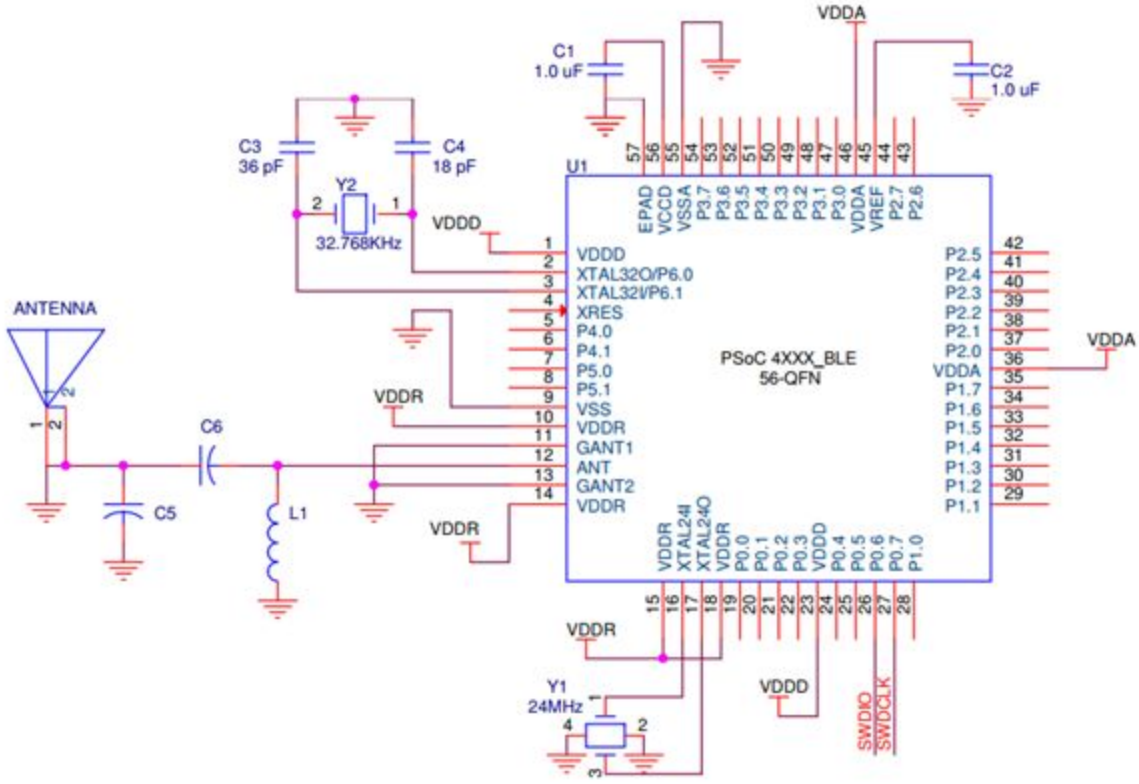


Figure 1. Schematics of BLE

Block Diagram

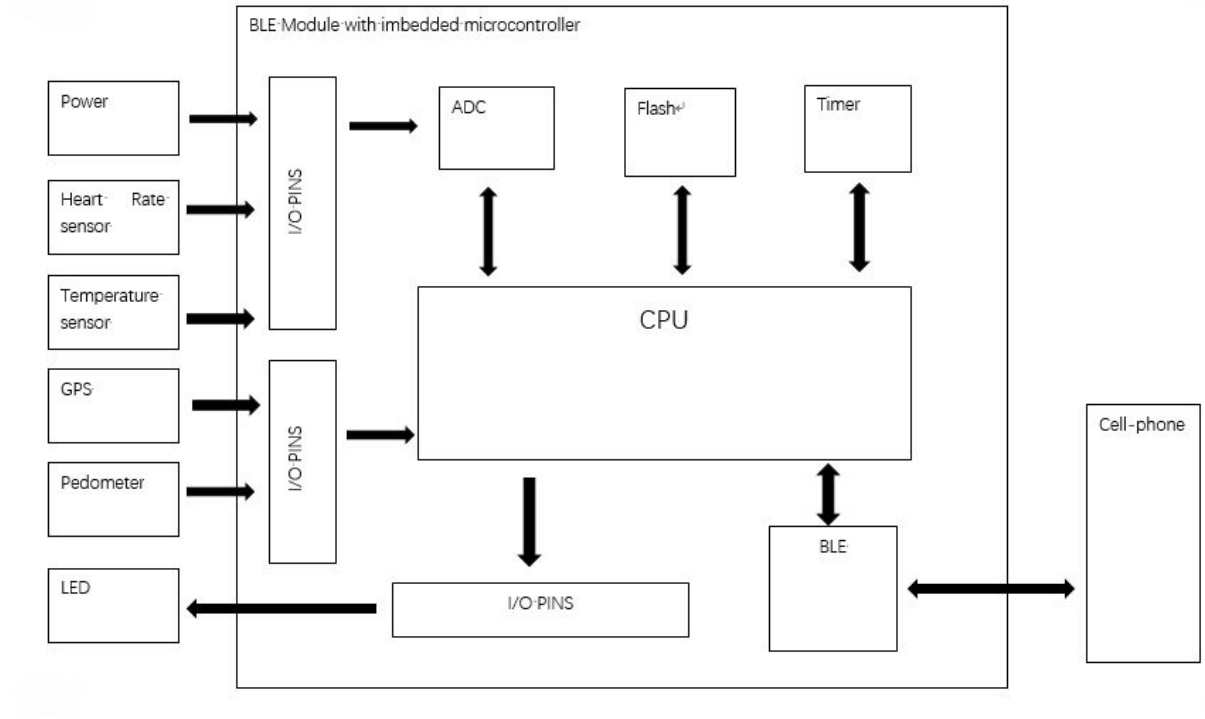


Figure 1. Block diagram of microcontroller and BLE.

In this block diagram, two analog sensors are connected with GPIOs and go through ADC. The other two sensors are digital so ADC is unnecessary. The flash memory can be code with the API provided by Cypress. It has a 128-kB memory and is enough for our maximum data size. The data size estimation will be discussed in later section. The timer will control the measurement frequency of the band. BLE will get read command sent by the cell-phone and send out the data required by the cell-phone. LEDs are connected directly to the I/O pins to indicated whether the device is in advertising, connected, disconnected, low power, etc.

Details into BLE

The BLE module, which is inside our armband in this case, is the peripheral during advertisement mode which is before the connection has been made. It will be a slave after the connection is done. The arm band advertise itself so that the central device, which is our

cell-phone can find it and make connection to it. It has multiple service such as heart rate and thermometer, etc. The service table is provided below.
For attributes in our BLE service, each data will be recorded as 2 bytes so that the length of each packet will be 16. This means that each packet contains one raw data from the sensor. For heart rate sensor measured at 30 sec intervals, each packet will contains the data within 30 sec. This is the same for temperature sensor and pedometer. For GPS, every two packet will contains the location within 30 sec. The data estimation table is provided bellow.
For our BLE module, the minimal advertising interval is 20ms and the maximum advertising interval is 30ms. In advertisement there will be local name of the device and UUID of the services.
For BLE security model, we are using mode 1 for security mode and unauthenticated pairing with encryption for security level. Before the connection is made, the two devices must be paired with each other first.

Table of BLE Services and Characteristics

Custom BLE Profile

Services

Battery Service
- I Battery Level
- I Battery Power State

Jogging Mode service
- I Measurement Frequency Change
- I Write Data to Flash

Heart Rate
- I Heart Rate Measurement
- I Heart Rate Max
- I Resting Heart Rate

Health Thermometer
- I Temperature Measurement
- I Temperature Celsius
- I Temperature Fahrenheit

Location and Navigation
- I LN Feature
- I Location and Speed

Get Past Data
- I Read Data from Flash
- I Delete Past when data are read

Alert Notification Service
- I Supported New Alert Category
- I New Alert

l Supported Unread Alert Category
l Unread Alert Status
l Alert Notification Control Point

Device Information

l Manufacturer Name String
l Model Number String

Table 1. BLE Service and characteristics

Memory and Data Transmission Estimation

	Heart Rate	Temperature	Speed	Location
Maximum Size per data (byte)	2	2	2	4
Maximum Frequency (times/minute)	2	2	2	2
Maximum Measured Hours (h)	6	6	6	6
Maximum data size(byte)	1440	1440	1440	2880
Sum(kB)				7.2

Table 4. Estimation of data size

Temperature Measurement

To process the signal from the temperature sensor, an ADC is used for achieve this purpose. The resolution of the ADC is 12 bits and the channel sample rate is 3030 SPS. The reference voltage is the internal 1.024V. The output voltage of the temperature sensor is proportional to the actual temperature in celsius. Temp(Celsius) = (Vout - 500)/10. This equation is provided by the datasheet of the temperature sensor. The ADC and MCU calculation is achieved by Cypress API.

Testing Result and ADC schematics:

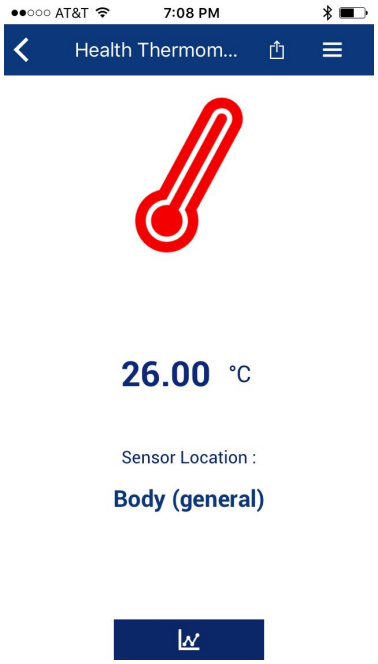


Figure 3. Testing Result of the sensor at room temperature

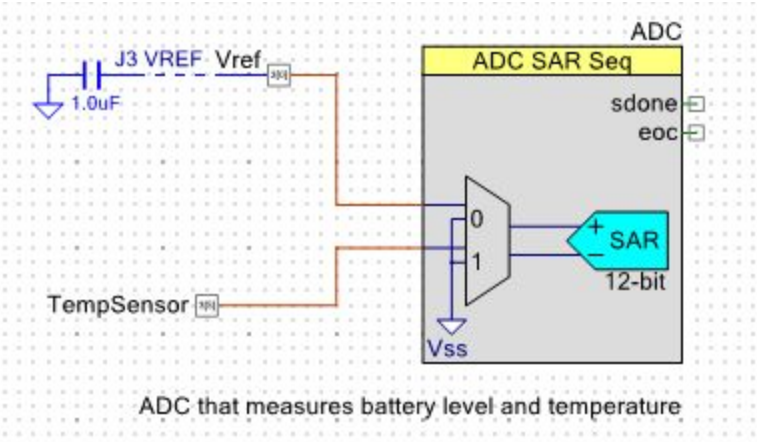


Figure 4. Schematics of ADC

Battery Level Measurement

Because the battery level is not linearly proportional to VDD. The analysis is divided into three parts (0-2v, 2-2.8v,2.8-3v), each part is treated as a linearly change. When VDD is between 0-2v, the battery level is considered to be at 0%. When VDD is between 2-2.8v, the battery level is proportional to VDD from 0 - 29%. When VDD is between 2.8-3v, the battery level is proportional to VDD from 29 - 100%.

In the circuit, the reference of ADC is set to VREF. The bypass is enabled for a short time (25 ms in this test) and the external capacitor is charged to VREF. Then the reference of the ADC to

VDD and measure the voltage on P3[0]. Then VDD is calculated as following: VDD = (1.024 * Full Scale Counts) / ADC Counts P3[0].

Test Result and ADC schematics:

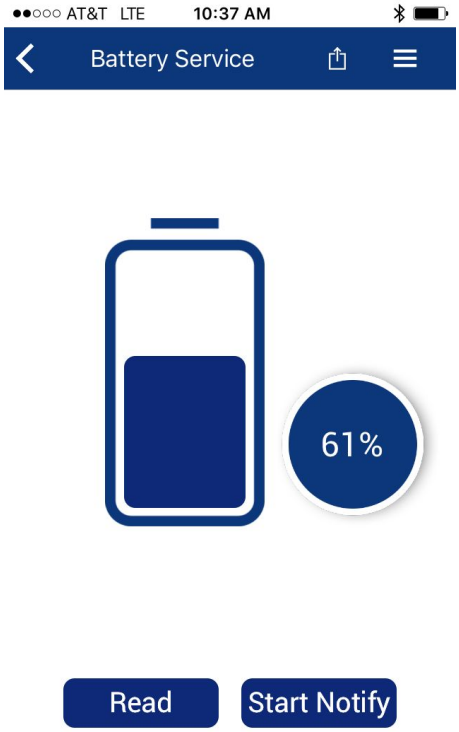


Figure 3. Test result of battery level when VDD = 2.89V

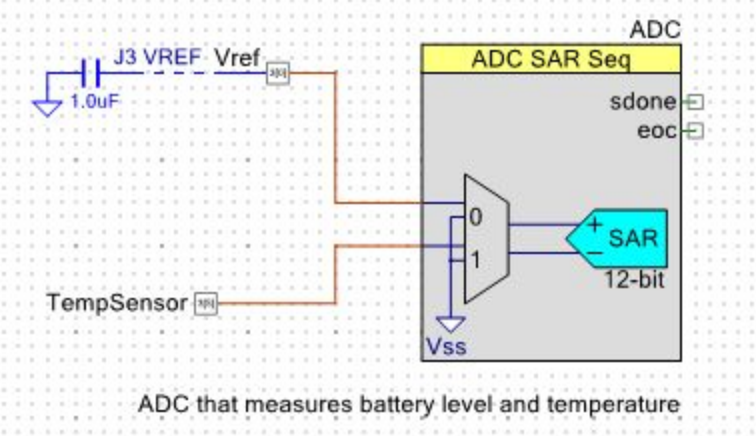


Figure 4. Schematics of ADC

Total Power Consumption Estimation

	IDD	Unit	VDD	Unit	Total Number of VDD on the chip	Total Power
Sleep Mode	1.3	uA	3.3	V	7	0.03mW
Hibernate Mode	150	nA	3.3	V	7	3.465uW
Working Mode	1.7	mA	3.3	V	7	39.27mW

Table 3. Estimation of total Power consumption

The BLE device will enter sleep mode after advertising period (30s) or after being disconnected by the cell-phone. The BLE device will enter hibernate mode if it is not waked up from the sleep mode after 30 sec. The sleep mode and hibernate mode can be waked up by an external switch.

Android Application

Technical overview

Our Android APP is the user-interface of our armband, it communicates with the microcontroller of our armband via Bluetooth Low Energy to read saved data and write measuring commands.

It drags raw data from two BLE characteristics that we designed, one contains GPS latitude, longitude and measured time in 12 bytes of space, the other contains temperature, heartrate, steps and current battery level in 5 bytes. After the reading of data, my app converting them into correct numbers using the functions I wrote according to different converting rules for each sensor that Junyan (Our microcontroller developer) and I designed together.

My App not only displays all the data that measured by our sensors, but also store all the synced data in the phone automatically using SQLite Database, I designed two databases one for GPS data with time and the other for temperature, heartrate and steps since user usually don't need other data when they want to read for GPS data.

In the database, each kind of data occupies one column. With data stored in data base, my App displays all the sensor measured data on several line graphs, and displays all the stored GPS coordinators on the Google Map API section as markers with time tag. On the map section, user can take a marked location into Google Map App if he/she have one on their phone, thus they can easily either get more information about this location or navigating to this location. This app will support API 23 and later version, which is Android 6.0 and newer version of android. The signal flow diagram of our APP is shown below.

Detailed subsystem block diagram

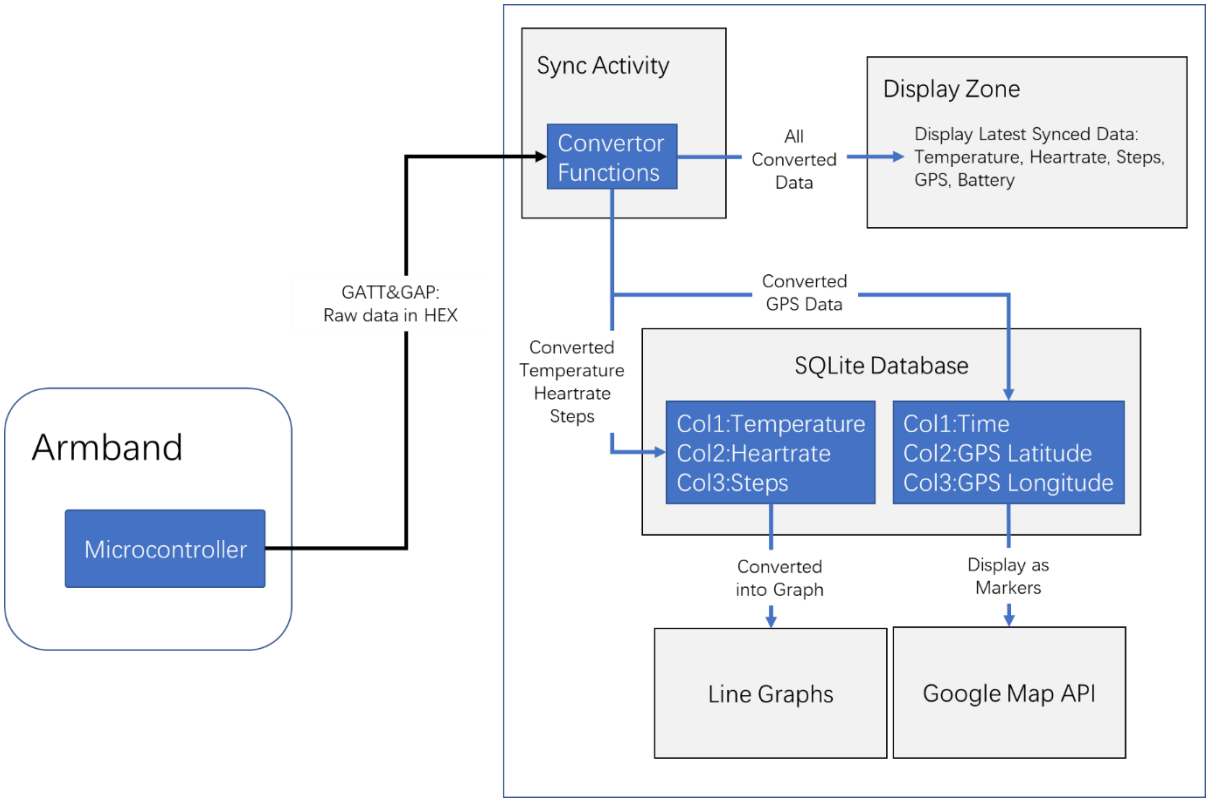


Figure 1, Block Diagram of Android App

Subsystem requirements

According to the team agreement and the developing of design through this semester, the

requirements of my Android App are listed below:

NO.	Requirement
1	Can search for BLE service with certain UUID and be able to pair with it.
2	Can drag raw data from BLE characteristics with certain UUID.
3	Can convert each kind of raw data to correct numbers according to specific rules designed by the microcontroller developer.
4	Can real-timely display all the converted data on screen while reading.
5	Can automatically save all the data into database.
6	Can display all the sensor data in database (except GPS) on line graphs respectively
7	Can display GPS data with time on Google Map API section as markers.

Table 1. Android APP subsystem requirement

As you can see from the block diagram in Figure 1, the Sync Activity of my App can pair with and read data from the microcontroller using BLE GATT and GAP. After the converting, the data will be both send to database and display on the screen simultaneously. Also, the block diagram also shows that all the data in the database will be displayed using either line graphs or map marks. Thus, all the requirements are meet.

Interfaces with others sub-systems

As described before, my app only communicates and interacts with microcontroller using Bluetooth Low Energy, in this process, the microcontroller performs as a peripheral and the cell-phone(App) is the central device. A custom profile was built and programmed by the microcontroller developer, which defined a new custom BLE service that contains two custom BLE characteristics. Both characteristics can be read and write via BLE.

The first characteristic is a 12-byte-long uint8 array that using for transferring the GPS coordinator and time, the latitude and longitude each using 5 bytes of length and the other 2 bytes are used for time, one for hour and one for minute. The second characteristic is using for transferring the data of temperature sensor, heartrate sensor, accelerometer and current battery level, which is a 5-byte-long uint8 array characteristic. In this characteristic, besides the step counter(accelerometer) data used 2 bytes of length, the other three types of data only using one byte each type.

The UUID of our custom BLE service and characteristics are all designed by ourselves and are 128 bit strings that contains letters and numbers. Specifically, our UUID for our self-designed BLE service and its characteristics are listed below:

Type	UUID
Custom Service	0000**8453**-0000-1000-8000-00805f9b34fb
Characteristic for GPS	0000**4B32**-0000-1000-8000-00805f9b34fb
Characteristic for Sensor and Battery	0000**4B31**-0000-1000-8000-00805f9b34fb

Table 2, UUIDs of custom designed BLE service and characteristics

As you can see from the table above, our UUID only different from each other by changing the fifth to eighth character. According to such design, I wrote a UUID API function in my APP that can let me define the target UUID for scanning, pairing or dragging data by simply input the four bits characters from fifth to eighth bit. The BLE advertising interval of our microcontroller is between 20ms and 30ms using Fast Advertising mode. Since we didn't define

the security protocol for our microcontroller, the connection has no password.

Design Evidence

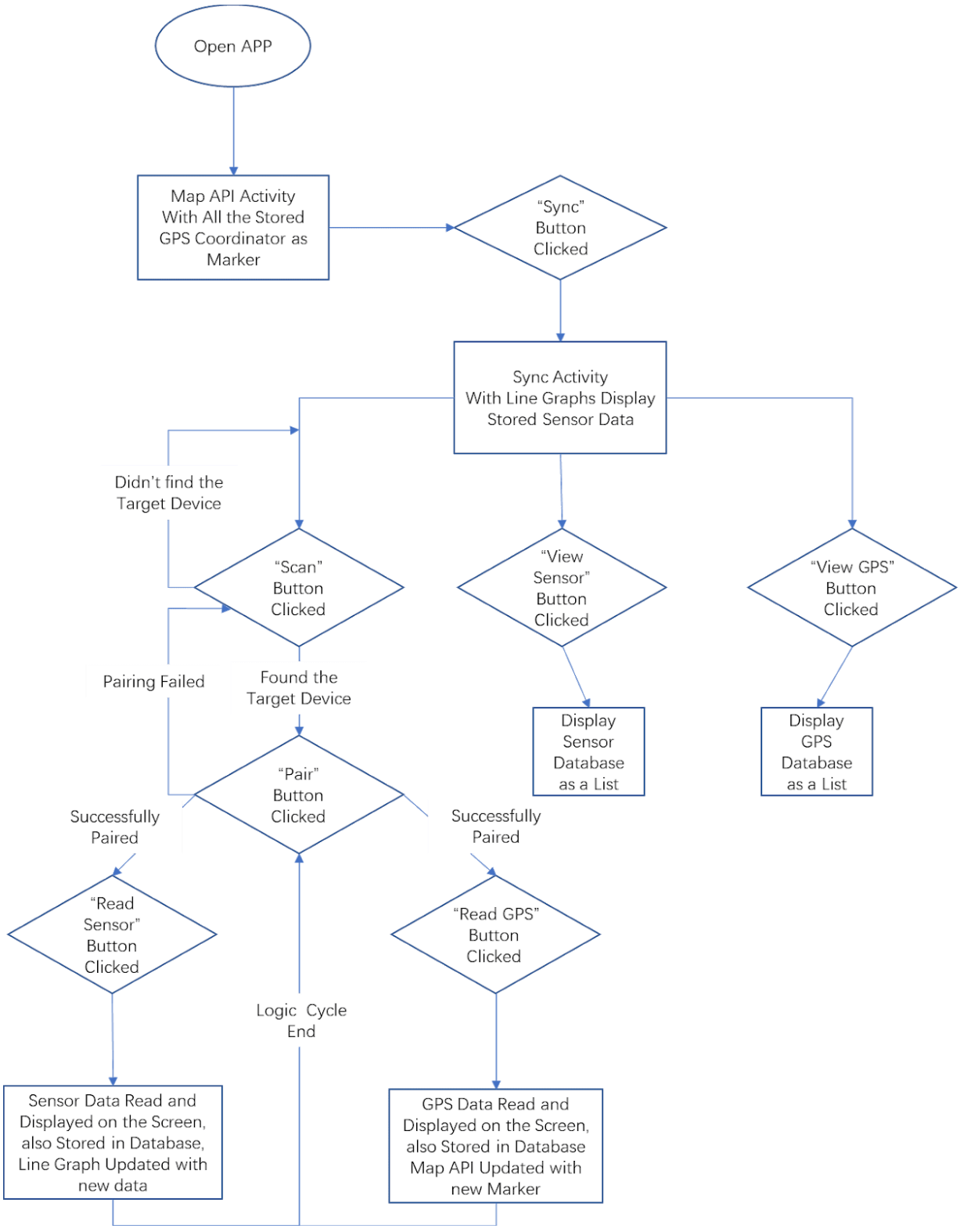


Figure 2, Android APP Logic Flow Diagrams

Partial Code for drag raw data from BLE Characteristic:

```
gatt.readCharacteristic(allCharacteristic);
gatt.readCharacteristic(gpsCharacteristic);
@Override
public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
    if (ALL_Char_UUID.equals(characteristic.getUuid().toString())) {
        byte[] allVal = characteristic.getValue();
        handleAll(allVal);
    }else if(GPS_Char_UUID.equals(characteristic.getUuid().toString())){
        byte [] gpsVal = characteristic.getValue();
        handleGPS(gpsVal);
    }
}

private void handleAll(byte[] val){
    double temperature = getCharTempValue(val);
    int Heartrate = getCharHRValue(val);
    int Step = getCharStepValue(val);
    int Batt = getCharBattValue(val);



    Tempdoub = temperature;
    HRint = Heartrate;
    Stepsint = Step;

    Temp = Double.toString(temperature);
    HR = Integer.toString(Heartrate);
    Steps = Integer.toString(Step);
    Battery = Integer.toString(Batt);

    mHandler.post(new Runnable() {
        @Override
        public void run() {
            showTempStr += "Temperature:"+Temp+"°
C\n"+"HR:"+HR+"bpm\n"+"Step:"+Steps+"steps\n"+"Battery:"+Battery+"%\n\n";
            temperTv.setText(showTempStr);
            AddData();
        }
    });
}
```

As you can see, all the conversion, printing, storing of data are operated on a callback function called *onCharacteristicRead,* which is a function that automatically operated when the app is trying to read any data from the microcontroller via BLE

Tests

To test my App's BLE communication functioning, I just simply let our microcontroller developer manually input some certain values of the sensors read on microcontroller, and then I operate my app to read the values via BLE characteristics and convert them into normal numbers, finally compare the printed result on the screen with the values that inputted by our microcontroller developer, if they are same, the BLE test is successful.

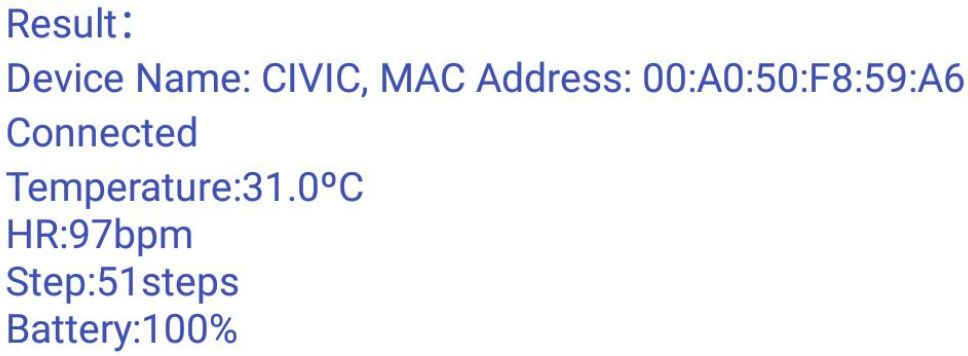


Figure 3, Data reading screenshot of our Android App

Then, to test whether the data storage is functioning, I've wrote two buttons that use for pulling up the data that stored in the database and display them as a list on the interface of the App. When finished reading from microcontroller, simply click the "View Sensor" button and see whether the data have been added to the list, if so, then the data storage is successfully working.

For my App, all the tests are successful, all the designed goal are achieved.

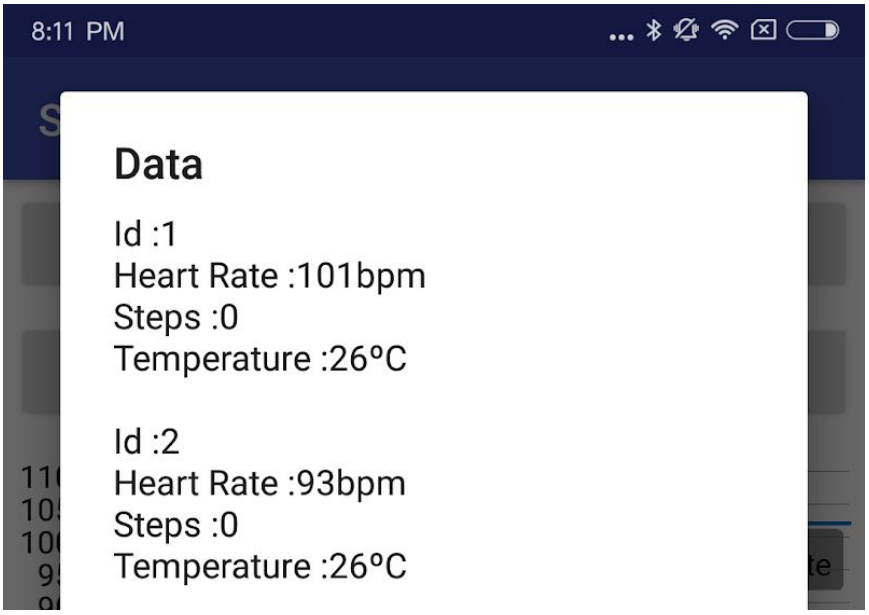


Figure 4, Data Storage List Screen Shot

Photo of prototypes

Since my subsystem is an Android App, I will just use the interface screenshots as my prototype photo.

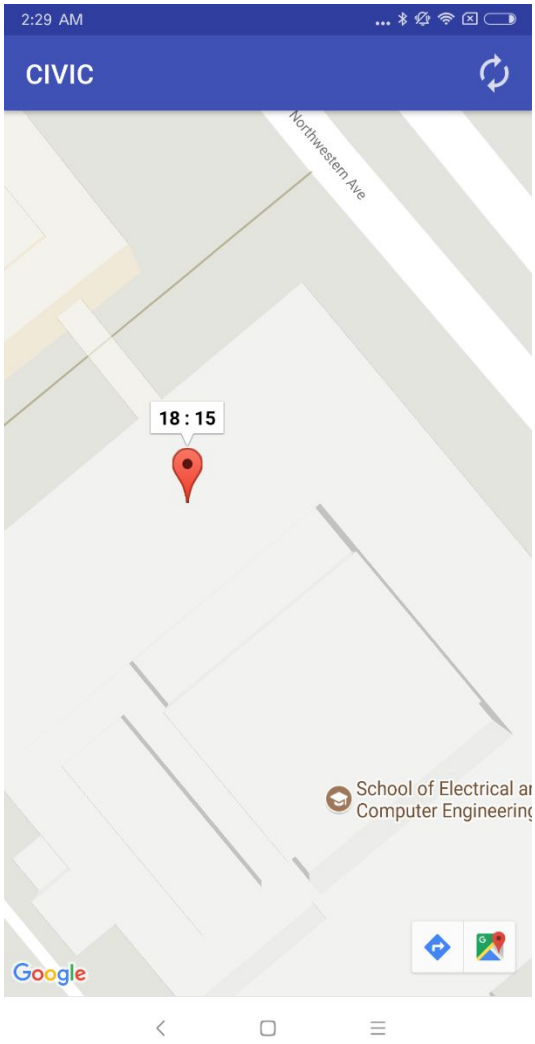


Figure 5, Map API Interface

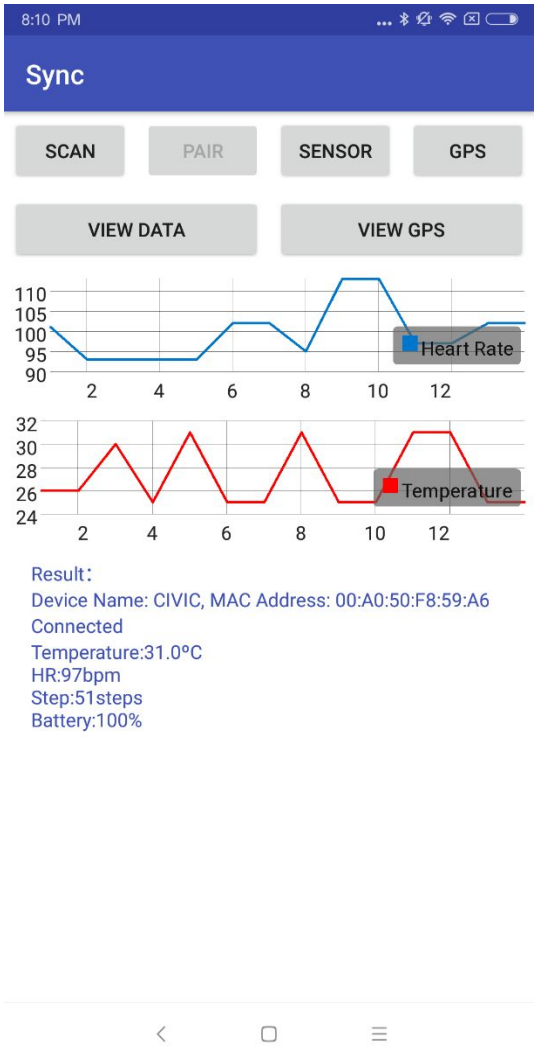


Figure 6, Sync Activity Interface

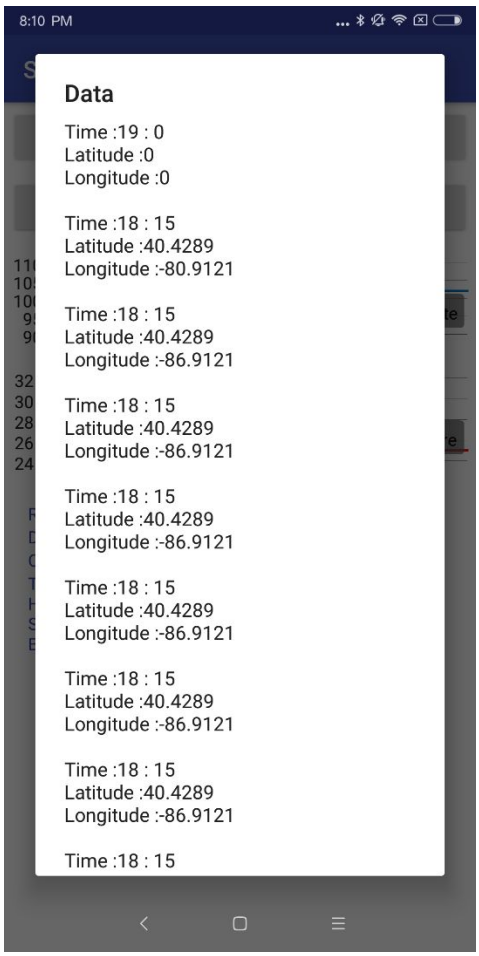


Figure 7, GPS Database In-App Display

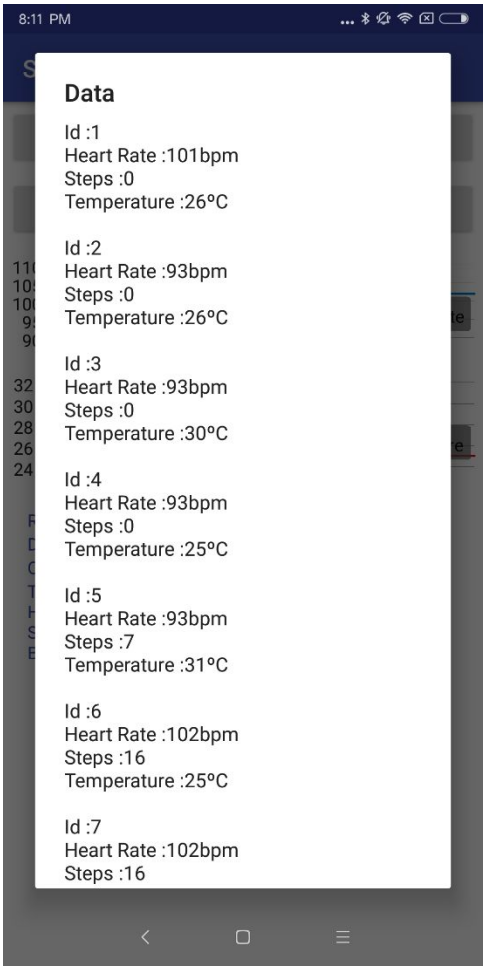


Figure 8, Sensor Database In-App Display

Power Subsystem Overview

The design of the fitness armband uses a rechargeable battery as a power source. The battery chosen (Adafruit Accessories Lithium Ion Polymer Battery 3.7V 500mAh) operates within 4.2V and 3.0V. While discharging, the battery has a nominal voltage of 3.7V. At a full discharge, the battery has 3.0V. The battery needs 4.2V in order to charge. Aside from the operational amplifiers of the heart rate monitor (which uses 5V), all the components of the armband use 3.3V. During normal operation the thermometer uses 6μA and has a peak value of 15μA. The accelerometer will vary within 3.3μA and 1.8μA, with a normal operation of 1.8μA. Generally speaking, these components have an idle power consumption of about 2.6μW and peak values of approximately 6μW. Neither of these values are large enough to be a concerning large source of power consumption.

In order to power all the components, the armband uses a power regulator (MAX 1701) that increases the battery voltage from 3.0V – 3.7V to 5V. The regulator then feeds 5V to the operational amplifiers of the heart rate monitor and then uses a low drop-out (LDO) regulator to decrease the voltage to 3.3V and power the remaining components in the board.

In order to charge the battery, the armband will be connected to wall outlet that uses a 5V and 2A adapter. To charge the battery, a charge manager controller (MCP73831/2) that decreases the voltage to 4.2V and the current to 500mA. The adapter will also connect to the controller and the battery through a microUSB connector.

Power Diagram

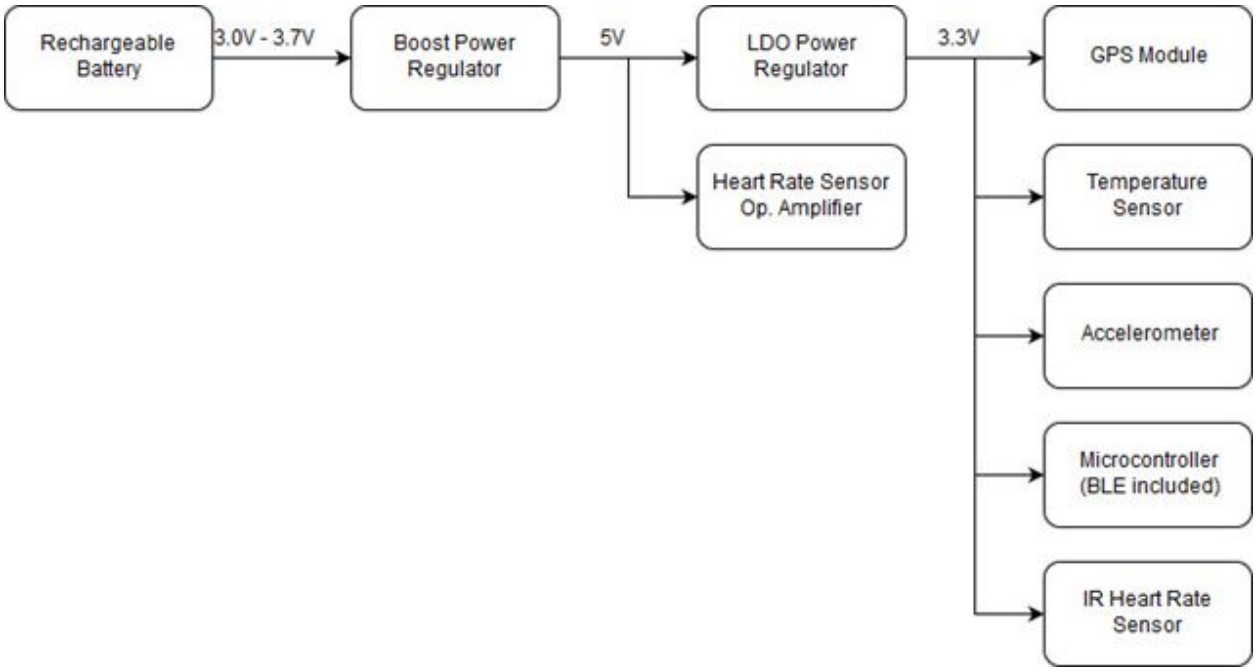


Figure 8: Power subsystem diagram

Power Requirements

As shown in Figure 12, the power subsystem requires three subsections in order to power the whole armband system: the 5V section, the 3.3V section and the charging (4.2V) section.

In order to increase the voltage to 5V, a high-efficiency MAX1701 DCDC converter is used. This converter has a maximum output current of 800mA. The armband has current levels well under this limit. The MAX 1701 uses a voltage divider with varying resistor values at the output in order to output a maximum voltage of 5.5V. Since the rechargeable battery only varies between 3.0V and 3.7V (as well as 4.2V when charging), the regulator can output constant 5V without variation. The 5V output is used to power the operational amplifiers of the heart rate monitor and then becomes the input of the 3.3V LDO. The datasheet suggests the use of a 10μH inductor to assure the internal N-channel switch is not saturated.

The LDO regulator uses the 5V output of the boost regulator in order to power the remaining components of the armband. The steady 5V from the MAX 1701 allows for a constant 3.3V output from the LDO.

In order to charge the battery, the armband may use a common wall outlet adapter that outputs 5V and 2A that then connects to the PCB through a microUSB connector. The output of the microUSB connector will then become a 5V / 2A input for the MCP73831/2 charge manager. The charge manager will then output 4.2V and 500mA to charge the battery.

The following table shows the power requirements of the whole armband system:

Component	**Max. Current**	**Typical Current**	**Voltage**	**Max. Power**	**Typical Power**
Thermometer	15μA	6μA	3.3V	49.5μW	19.8μW
Accelerometer	3.3μA	1.8μA	3.3V	10.89μW	5.94μW
Heart rate IR sensor	100μA	100μA	5V	500μW	500μW
Operational amplifier	3mA	1.5mA	5V	15mW	7.5mA
GPS module	37mA	37mA	3.3V	122.1mW	122.1mA
Microcontroller / BLE module	1.7mA	1.7mA	3.3V	5.61mW	5.61mA
Total	41.81mA	40.30mA	-	142.36mW	135.73mW

Figure 9: Overall power budget

Power Subsystem Interfaces

The MAX 1701 uses a voltage divider to interact between the rechargeable battery and the 3.3V LDO. The following formula is used to establish the values of the resistors used in the voltage divider at the output of the 5V regulator:

$$R_1 = R_2\left(\frac{V_{out}}{V_{FB}} - 1\right)$$

Where V_{FB} = 1.23V (value provided by the datasheet of the MAX1701) and R_2 = 270kΩ (or less). In order to output the desired 5V necessary to power the operational amplifiers of the heart rate monitor and the 3.3V LDO, the regulator needs an 827.560MΩ resistor for R_1.

The advantage of using this regulator is constant 5V even when the battery output is at full discharge (3.0V) and fully charged (measured at 3.8V during lab). This regulator is also a reasonable size that can be easily tested within a breadboard for prototyping and a PCB for the final delivered project. This regulator is also efficient enough to support all the components used in the arm band.

Power Design and Evidence

Below is the schematic for the MAX 1701 boost regulator section of the power subsystem:

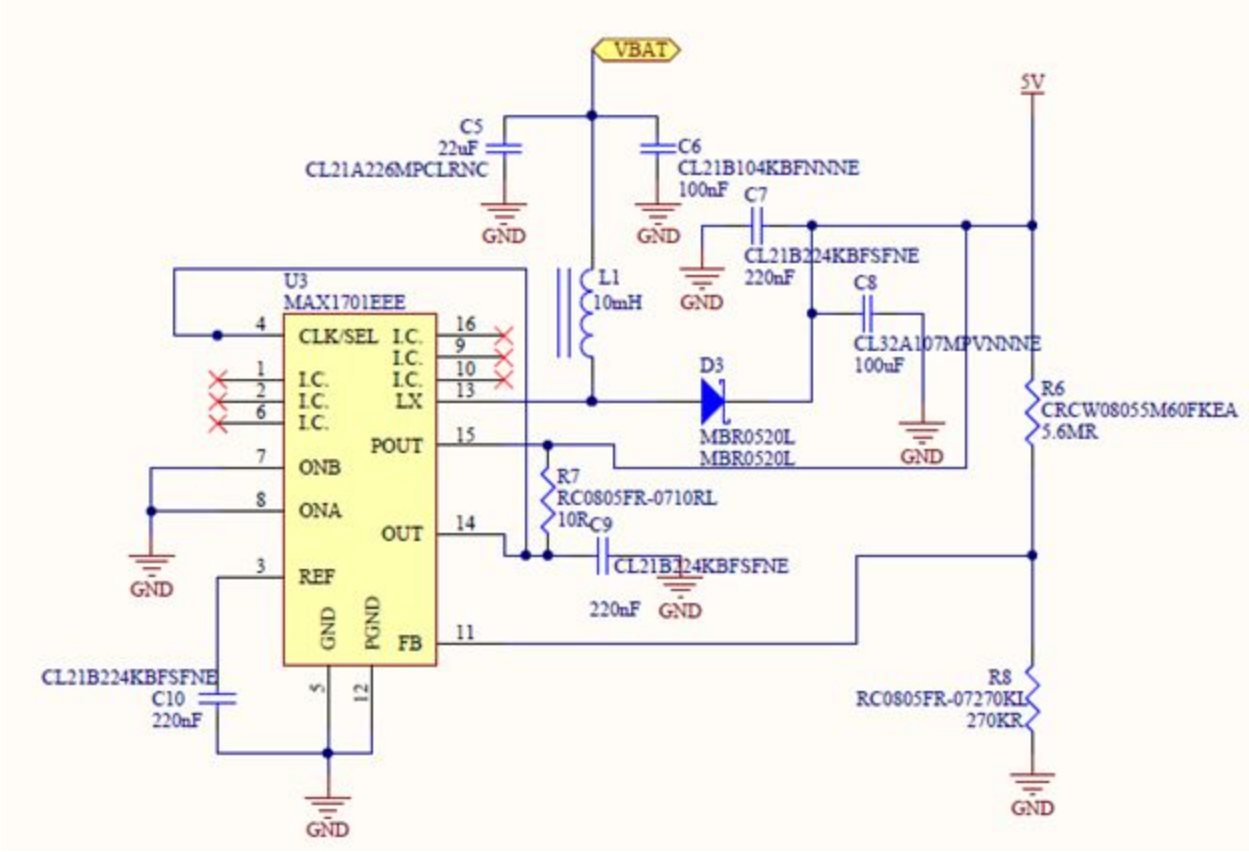


Figure 10: 5V regulator schematic

While gathering data with the MAX1701, the values of the voltage divider resistors was changed several times in order to check that a more accurate output voltage could be created. Aside from a few miscalculations during the initial prototyping, it became evident that the best result came from assigning 27kΩ to R_8 and 75kΩ to R_6. When these values were tested on the printed PCB, the output voltage was 4.1V, as seen in the screen below:

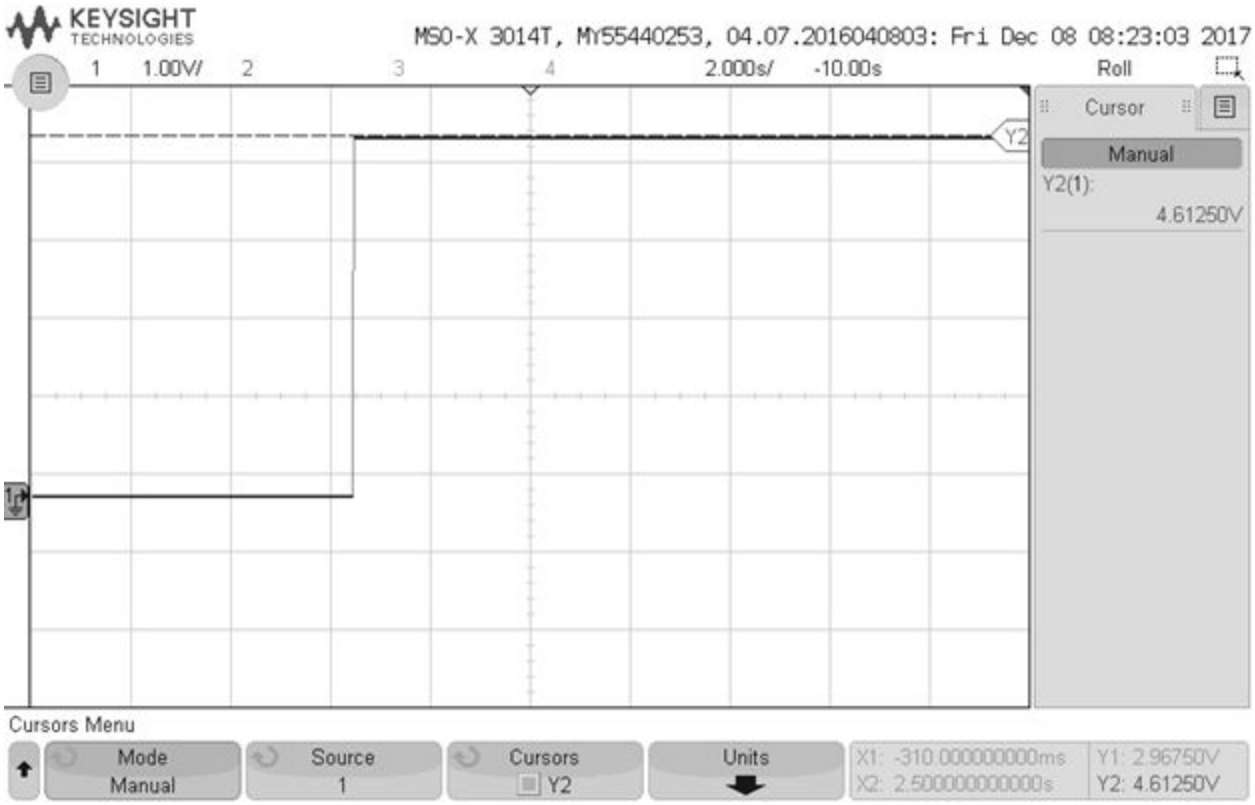


Figure 11: MAX 1701 oscilloscope output (varying voltage input)

Additionally, the intended schematic for the 3.3V LDO was the following:

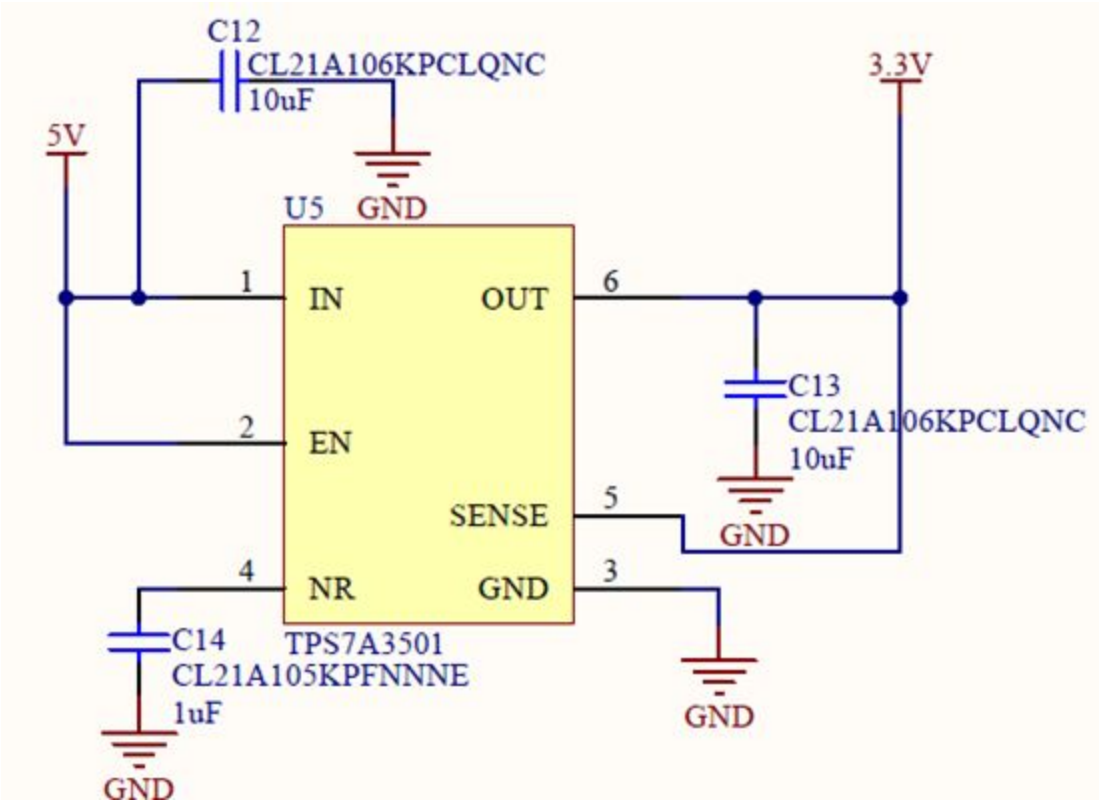


Figure 12: 3.3V LDO schematic

While trying to prototype the LDO, it became obvious that the part number chosen was in fact actually an LDO regulator, and therefore the voltage at the output was 3.8V, instead of 3.3V. For this reason, this section of the power supply did not make it to the printed PCB. Yet, in order to power 3.3V components, another MAX1701 was used. If no voltage divider is used are the output of the regulator, it outputs 3.3V. This can is shown in the screen below:

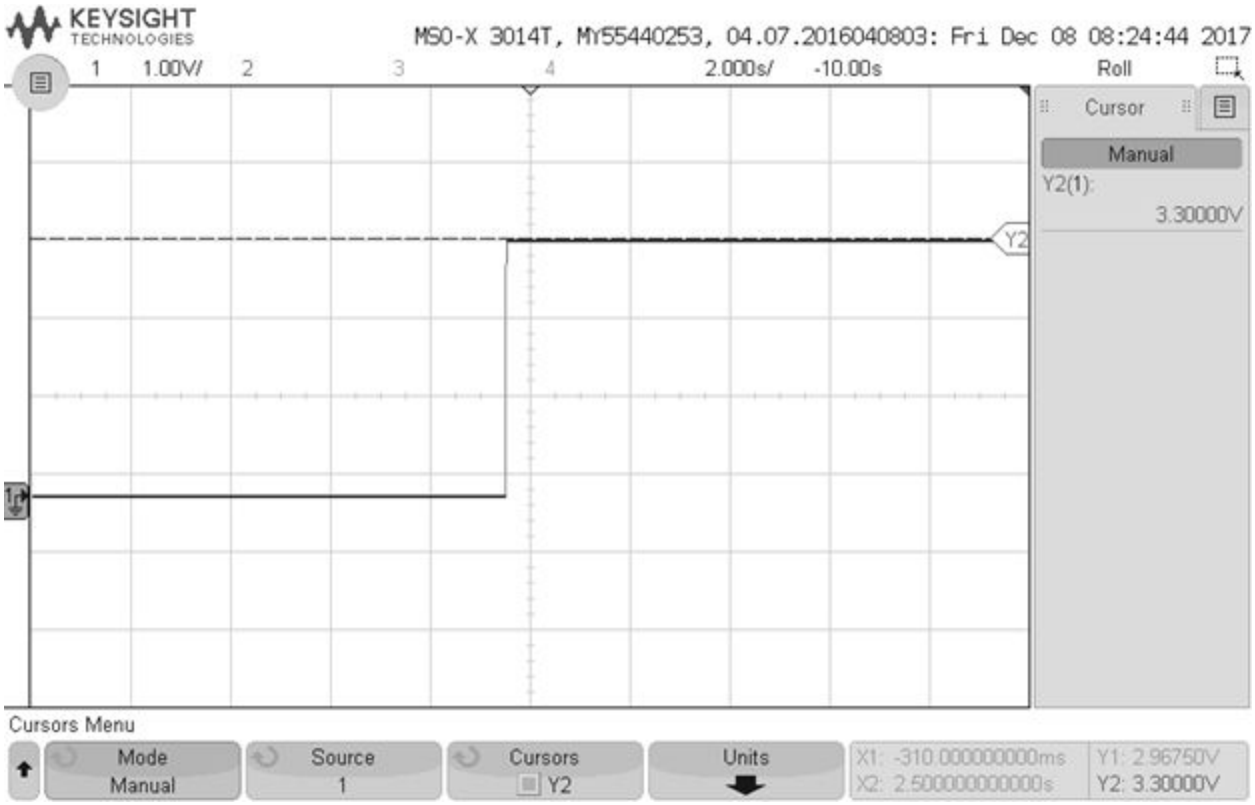


Figure 13: MAX 1701 oscilloscope 3.3V output (varying input voltage)

Figure 11 and 13 both demonstrate that the regulator outputs a steady output even when the input of the regulator varies.

The last intended section of the power subsystem is shown in the schematic below:

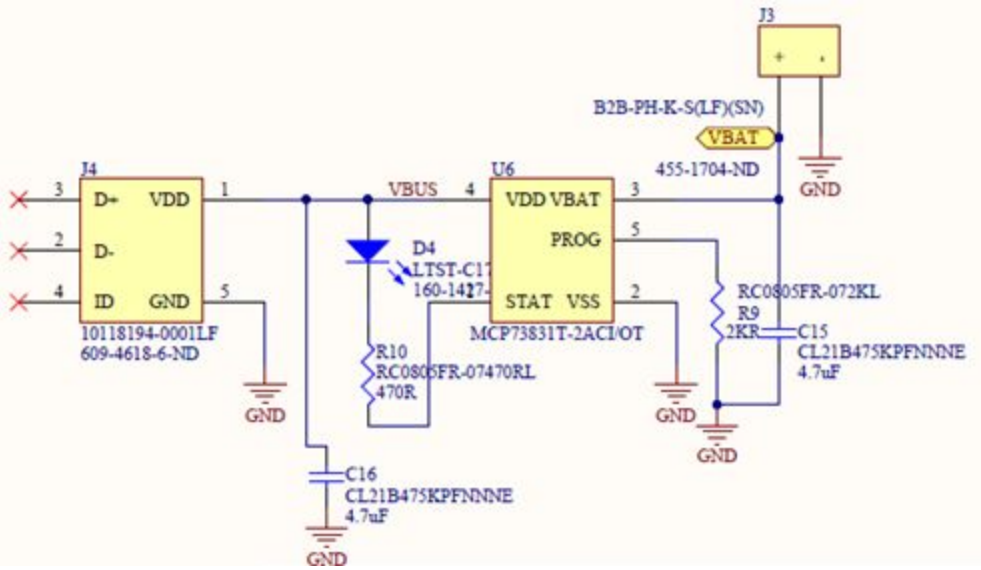


Figure 14: Battery charging schematic

Where J4 is a microUSB connector and J3 is a connector that matches the header of the rechargeable battery. Unfortunately, due to lack of time before the final demonstration, this part of the subsystem did not get tested on the printed PCB.

Power Tests

In this subsystem, the three sections were addressed separately.

To test the 5V regulator, the components necessary were soldered onto the printed PCB and wires were soldered to the power connector pads in order to have a power source. A power supply at the lab bench was used, in order to be able to limit the current, since the MAX 1701 is really sensitive and tends to get damaged with too much current. Lastly, the DMM in the lab was used in order to measure voltage between any pad at fed 5V to components and ground.

To test the 3.3V regulator, a similar procedure was used. Yet, the voltage measurement from the DMM was between any 3.3V pad and ground. After integrating the LDO chip to the PCB, it became obvious that 3.8V were measured due to the fact that the LDO chip was not actually the expected chip. To resolve this, another printed PCB was used and a MAX1701 was used (without a voltage divider) to output 3.3V instead.

Lastly, due to lack of time, the last section of this subsystem (battery charging) was not tested, unfortunately.

Power Prototypes

The prototype used to test the 5V (which actually output 4.6V) MAX1701 regulator is shown below:

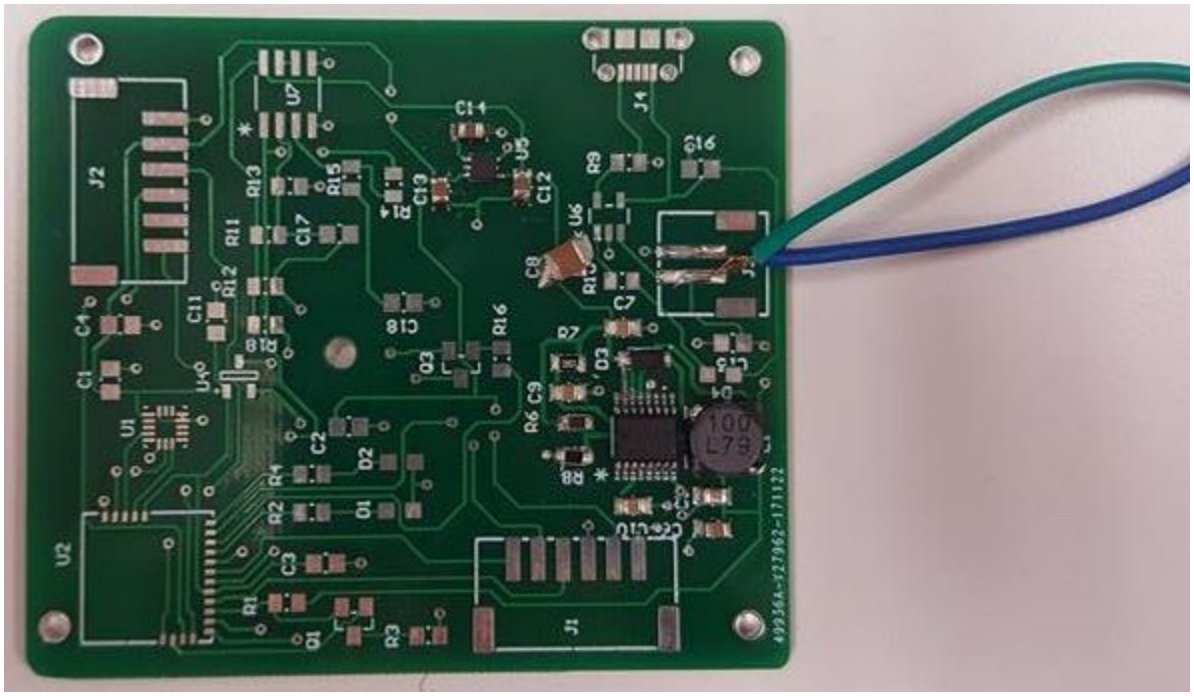


Figure 15: 5V MAX 1701 and 3.3V LDO prototype

The 3.3V LDO is also pictured above, although the main purpose of the prototype was to test the 5V section of the power subsystem. The prototype used to test the 3.3V regulator is shown below:

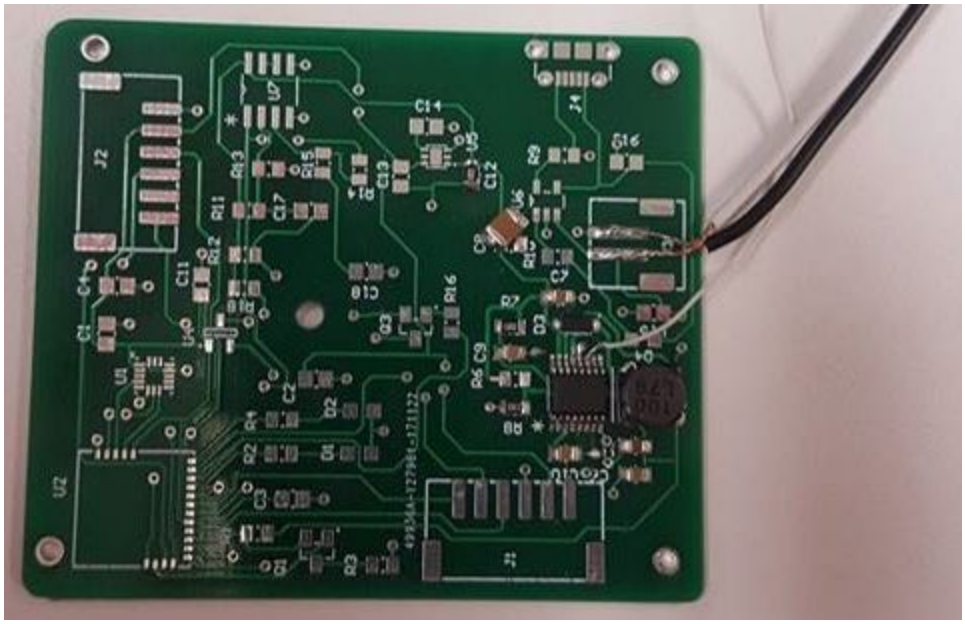


Figure 16: MAX 1701 3.3V prototype

Thermometer / Accelerometer Subsystem Overview

In order to count steps, the armband uses a three axis digital output accelerometer (ADXL 362). The accelerometer communicates with the microcontroller through SPI communication and uses an algorithm to determine if the armband is moving or stationary. The accelerometer measures data with a frequency of 100Hz and constantly updates step counting when the microcontroller is gathering data. In order to use the data of the axis that has the greatest variation in data, the algorithm squares the values of all axes, adds them up and then takes the square root of the resulting value. By using the magnitude of all the axes, the armband can also smooth out any minor deviations in data that might be mistaken as steps. The algorithm also needs a threshold (defined by the user) that can be used to determine if a deviation in the magnitude is great enough to be counted as a step. Once data has been gathered from the accelerometer and interpreted by the microcontroller, the user app will update when the user connects with the Android app through Bluetooth Low Energy (BLE).

The armband also uses an analog linear active thermistor (MCP9700) in order to gather temperature data from the environment. The thermal sensor can detect temperatures ranging between -40°C and +150°C. The thermometer communicates with the microcontroller through ADC (without the need of any amplifiers) and the microcontroller uses a simple transfer function provided by the datasheet to interpret the output voltage of the thermometer into temperatures into degrees Celsius. The microcontroller will then transfer the gathered data to the Android app and show the resulting temperature data in degrees Fahrenheit after connecting through BLE. Within the overall armband system, this subsystem requires the least amount of power, with negligible current and power (in microwatts) consumption.

Thermometer / Accelerometer Diagram

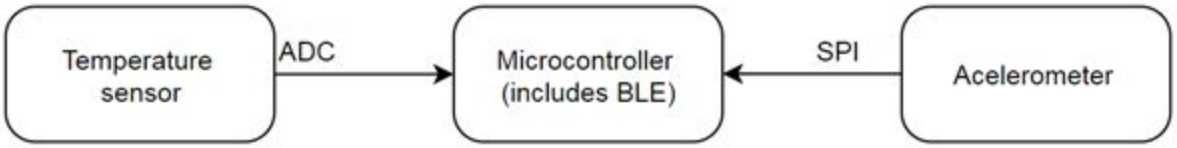


Figure 17: Thermometer and accelerometer subsystem diagram

Thermometer / Accelerometer Requirements

The thermometer selected requires a 3.3V power source and outputs a voltage that varies with changes in temperature. The maximum voltage for this IC is 5.5V and the minimum is 2.3V. The thermometer has a typical current of 6μA and a maximum current of 12μA. The voltage output then becomes the input for an ADC I/O pin of the microcontroller.

The accelerometer typically requires 3.3V and communicates with the microcontroller through an SPI protocol. This IC has a minimum voltage requirement of 1.71V and a maximum of 3.6V. The microcontroller then scans for data with a 100Hz frequency and uses an algorithm to interpret the data from all three axes to determine when steps are taken.

Thermometer / Accelerometer Interfaces

The interfaces of this subsystem consists of the communication protocols needed to communicate with the microcontroller used in the armband. The thermometer IC used only needs an ADC input to read the voltage output and calculate temperature with the following transfer function provided by the datasheet:

$$V_{out} = T_C * T_A + V_{0°C}$$

Where T_A is the ambient temperature (in degrees Celsius), T_C = 10mV/^oC and $V_{0\,^oC}$ = 500mV.

In order to interact with the microcontroller, the accelerometer uses SPI protocol. This protocol requires that the microcontroller first send the register address for the x-axis data (which is the axis with the highest priority), followed by the read command register address. The accelerometer will then return six 8-bit values to the microcontroller: two 8-bit values for each axis. These 8-bit values can then be assigned to three variables, one for every axis. The microcontroller can assign 8-bit to the x-axis, shift the values to accommodate the second 8-bits of the same axis, and then repeat this for the y- and z- axis (these values are returned to the microcontroller by priority). The microcontroller may then use these values in the algorithm created to count steps.

Thermometer / Accelerometer Design and Evidence

Since the thermometer is an analog device, its output is a voltage value that varies with differences in temperature. To test that these values are accurate, we connected the output of the thermometer to a DMM and measured the output voltage. In the table below, a few data points have been gathered, entered into the transfer function provided by the datasheet and then compared with values obtained with a digital thermometer available in the Senior Design lab:

V_{OUT} (V)	**T_A (°C)**	**Digital read (°F)**	**Digital read (°C)**
0.7468	24.68	74.5	76.424
0.7446	24.46	75.0	76.028
0.7561	25.61	79.0	78.098
0.7250	22.50	71.5	72.50

Figure 18: Thermometer voltage output and test comparison

For convenience, the reads from the digital thermometer in the lab has been converted to degrees Celsius in order to aid in the comparison of the values. In most of the reads gathered, the accuracy of the thermometer readings are within one or two degrees Celsius, which is within the accuracy that the CIVIC team was trying to reach.

Additionally, the schematic of the thermometer is below:

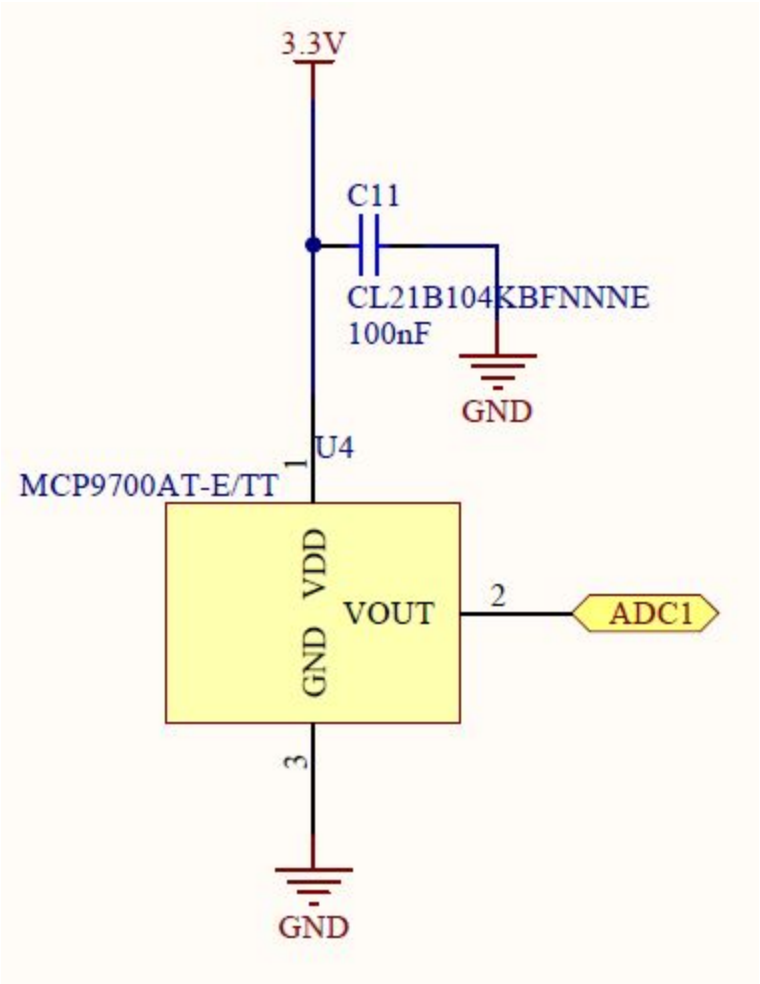


Figure 19: Thermometer schematic

The schematic for the accelerometer section of this subsystem is below:

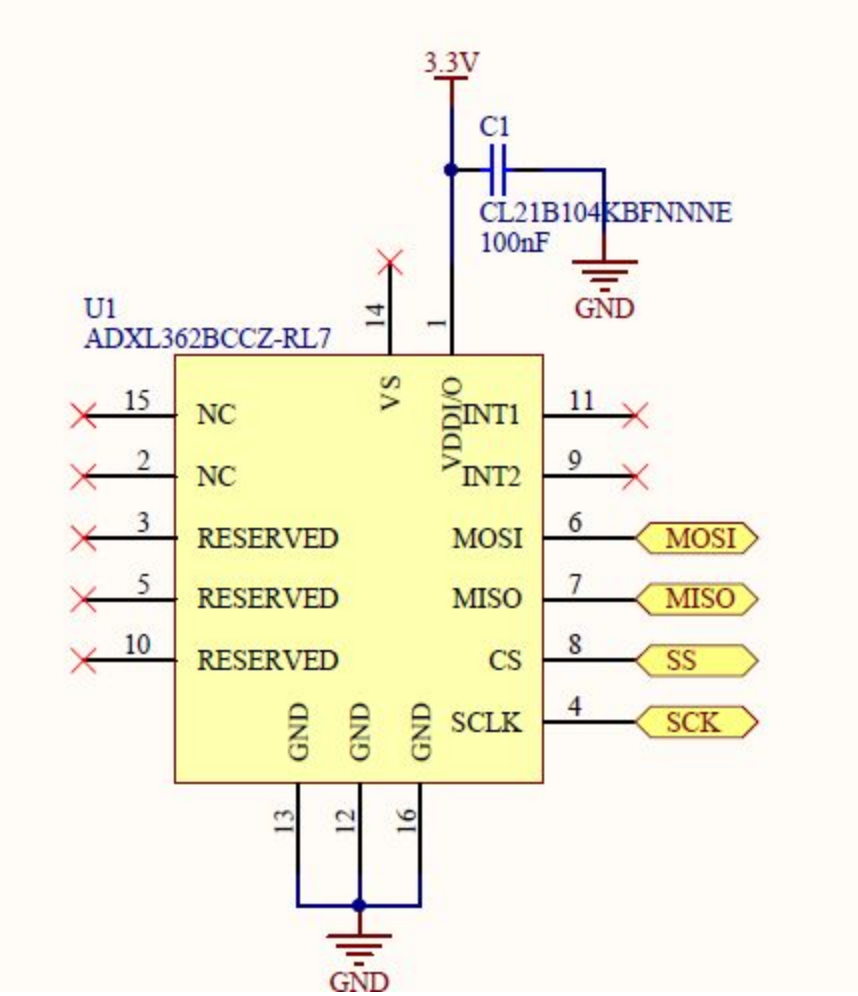


Figure 19: Accelerometer schematic

Where the four ports on pins 4 – 6 are the SPI communication pins that connect to the microcontroller. For prototyping, the accelerometer was initially coded with an Arduino in order to establish an algorithm to count steps first that could then be transferred to the microcontroller used in the armband. When the accelerometer is sitting on a flat surface the data from all three axes deviates quite a bit, as shown below.

XDATA: -200	YDATA: 79	ZDATA: 1363	TDATA: 259
XDATA: -202	YDATA: 83	ZDATA: 1359	TDATA: 259
XDATA: -202	YDATA: 83	ZDATA: 1366	TDATA: 259
XDATA: -205	YDATA: 80	ZDATA: 1363	TDATA: 259
XDATA: -202	YDATA: 87	ZDATA: 1366	TDATA: 259
XDATA: -199	YDATA: 83	ZDATA: 1367	TDATA: 262
XDATA: -200	YDATA: 81	ZDATA: 1367	TDATA: 259
XDATA: -200	YDATA: 80	ZDATA: 1368	TDATA: 259
XDATA: -200	YDATA: 79	ZDATA: 1363	TDATA: 259
XDATA: -200	YDATA: 83	ZDATA: 1360	TDATA: 262
XDATA: -205	YDATA: 78	ZDATA: 1363	TDATA: 255
XDATA: -205	YDATA: 81	ZDATA: 1359	TDATA: 255
XDATA: -199	YDATA: 80	ZDATA: 1368	TDATA: 255
XDATA: -200	YDATA: 79	ZDATA: 1368	TDATA: 259
XDATA: -201	YDATA: 86	ZDATA: 1367	TDATA: 255
XDATA: -200	YDATA: 79	ZDATA: 1363	TDATA: 259
XDATA: -202	YDATA: 80	ZDATA: 1359	TDATA: 259
XDATA: -199	YDATA: 79	ZDATA: 1363	TDATA: 255
XDATA: -200	YDATA: 79	ZDATA: 1366	TDATA: 259
XDATA: -201	YDATA: 79	ZDATA: 1363	TDATA: 259
XDATA: -201	YDATA: 79	ZDATA: 1359	TDATA: 255
XDATA: -200	YDATA: 81	ZDATA: 1363	TDATA: 259
XDATA: -201	YDATA: 83	ZDATA: 1363	TDATA: 259
XDATA: -193	YDATA: 79	ZDATA: 1367	TDATA: 255
XDATA: -201	YDATA: 81	ZDATA: 1363	TDATA: 255
XDATA: -199	YDATA: 79	ZDATA: 1366	TDATA: 259
XDATA: -200	YDATA: 79	ZDATA: 1360	TDATA: 259
XDATA: -200	YDATA: 78	ZDATA: 1359	TDATA: 259
XDATA: -200	YDATA: 81	ZDATA: 1366	TDATA: 257
XDATA: -197	YDATA: 76	ZDATA: 1358	TDATA: 255
XDATA: -200	YDATA: 80	ZDATA: 1359	TDATA: 259
XDATA: -200	YDATA: 79	ZDATA: 1372	TDATA: 255

Figure 20: Data read from the ADXL 362 through SPI

In figure 20, XDATA, YDATA, ZDATA and TDATA represent the x- axis, y-axis, z-axis and temperature data. In order to smooth out these deviations, the algorithm squares the value in every axis, adds them together and then takes the square root of this value (TDATA will be ignored within the algorithm, since it is not affected by movement of the accelerometer). This value (called MAGNITUDE) increases with spikes in any of the axes but deviates less than the data taken directly from the accelerometer.

```
XDATA: -200     YDATA: 79       ZDATA: 1368     MAGNITUDE: 1384
XDATA: -200     YDATA: 79       ZDATA: 1363     MAGNITUDE: 1379
XDATA: -200     YDATA: 79       ZDATA: 1363     MAGNITUDE: 1379
XDATA: -199     YDATA: 79       ZDATA: 1366     MAGNITUDE: 1382
XDATA: -200     YDATA: 79       ZDATA: 1367     MAGNITUDE: 1383
XDATA: -201     YDATA: 79       ZDATA: 1366     MAGNITUDE: 1382
XDATA: -194     YDATA: 79       ZDATA: 1363     MAGNITUDE: 1379
XDATA: -202     YDATA: 83       ZDATA: 1361     MAGNITUDE: 1378
XDATA: -200     YDATA: 79       ZDATA: 1363     MAGNITUDE: 1379
XDATA: -200     YDATA: 79       ZDATA: 1361     MAGNITUDE: 1377
XDATA: -199     YDATA: 80       ZDATA: 1367     MAGNITUDE: 1383
XDATA: -196     YDATA: 79       ZDATA: 1358     MAGNITUDE: 1374
XDATA: -200     YDATA: 79       ZDATA: 1359     MAGNITUDE: 1375
XDATA: -200     YDATA: 80       ZDATA: 1369     MAGNITUDE: 1385
XDATA: -202     YDATA: 79       ZDATA: 1363     MAGNITUDE: 1380
XDATA: -200     YDATA: 79       ZDATA: 1359     MAGNITUDE: 1375
XDATA: -202     YDATA: 80       ZDATA: 1363     MAGNITUDE: 1380
XDATA: -208     YDATA: 81       ZDATA: 1363     MAGNITUDE: 1381
XDATA: -207     YDATA: 71       ZDATA: 1368     MAGNITUDE: 1385
XDATA: -205     YDATA: 78       ZDATA: 1367     MAGNITUDE: 1384
XDATA: -200     YDATA: 76       ZDATA: 1361     MAGNITUDE: 1377
XDATA: -208     YDATA: 76       ZDATA: 1374     MAGNITUDE: 1391
XDATA: -205     YDATA: 71       ZDATA: 1351     MAGNITUDE: 1368
```

Figure 21: Data from x, y, z axes and calculated magnitude

From the data displayed in figure 21, it's possible to notice that the magnitude when the accelerometer is sitting still is approximately around 1,375 ± 10. When walking, values will increase past 1,375 and return to this value when still again, or when a "step" is complete. In order to start counting steps, a threshold of 1,400 is established to signal a spike in the data. When the magnitude values cross the threshold value, the algorithm raises a flag and then lowers it when the threshold has been crossed again (magnitude values are now be less than the threshold value). The algorithm counts a "step" after the flag has been raised *and* lowered.

```
Start of data read

XDATA: -991     YDATA: 1840     ZDATA: -32768   MAGNITUDE: 32834        FLAG: 1 FLAG: 1 STEP: 0
XDATA: 48       YDATA: 16       ZDATA: -32064   MAGNITUDE: 32064        FLAG: 1 FLAG: 1 STEP: 0
XDATA: -975     YDATA: 304      ZDATA: 128      MAGNITUDE: 1029 FLAG: 1 FLAG: 0 STEP: 1
XDATA: -967     YDATA: 16       ZDATA: 768      MAGNITUDE: 1234 FLAG: 0 FLAG: 0 STEP: 1
XDATA: 48       YDATA: 304      ZDATA: 0        MAGNITUDE: 307  FLAG: 0 FLAG: 0 STEP: 1
XDATA: -967     YDATA: 272      ZDATA: -31808   MAGNITUDE: 31823        FLAG: 1 FLAG: 1 STEP: 1
XDATA: -975     YDATA: 0        ZDATA: -32544   MAGNITUDE: 32558        FLAG: 1 FLAG: 1 STEP: 1
XDATA: -975     YDATA: 0        ZDATA: 240      MAGNITUDE: 1004 FLAG: 1 FLAG: 0 STEP: 2
XDATA: -975     YDATA: 252      ZDATA: 176      MAGNITUDE: 1022 FLAG: 0 FLAG: 0 STEP: 2
XDATA: -975     YDATA: 256      ZDATA: -32768   MAGNITUDE: 32783        FLAG: 1 FLAG: 1 STEP: 2
XDATA: -985     YDATA: 560      ZDATA: -32256   MAGNITUDE: 32275        FLAG: 1 FLAG: 1 STEP: 2
XDATA: -975     YDATA: 512      ZDATA: -32000   MAGNITUDE: 32018        FLAG: 1 FLAG: 1 STEP: 2
XDATA: -975     YDATA: 304      ZDATA: 0        MAGNITUDE: 1021 FLAG: 1 FLAG: 0 STEP: 3
XDATA: -977     YDATA: 768      ZDATA: -32128   MAGNITUDE: 32152        FLAG: 1 FLAG: 1 STEP: 3
XDATA: -975     YDATA: 304      ZDATA: -31808   MAGNITUDE: 31824        FLAG: 1 FLAG: 1 STEP: 3
XDATA: -963     YDATA: 16       ZDATA: -32512   MAGNITUDE: 32526        FLAG: 1 FLAG: 1 STEP: 3
XDATA: 48       YDATA: 32       ZDATA: -32064   MAGNITUDE: 32064        FLAG: 1 FLAG: 1 STEP: 3
XDATA: -975     YDATA: 560      ZDATA: -32000   MAGNITUDE: 32019        FLAG: 1 FLAG: 1 STEP: 3
XDATA: -975     YDATA: 16       ZDATA: -32320   MAGNITUDE: 32334        FLAG: 1 FLAG: 1 STEP: 3
XDATA: -963     YDATA: 256      ZDATA: -32320   MAGNITUDE: 32335        FLAG: 1 FLAG: 1 STEP: 3
XDATA: -2045    YDATA: 1216     ZDATA: -32768   MAGNITUDE: 32854        FLAG: 1 FLAG: 1 STEP: 3
XDATA: -975     YDATA: 64       ZDATA: -31808   MAGNITUDE: 31823        FLAG: 1 FLAG: 1 STEP: 3
XDATA: -991     YDATA: 656      ZDATA: -32000   MAGNITUDE: 32022        FLAG: 1 FLAG: 1 STEP: 3
XDATA: 32       YDATA: 768      ZDATA: 128      MAGNITUDE: 779  FLAG: 1 FLAG: 0 STEP: 4
XDATA: 32       YDATA: 96       ZDATA: 256      MAGNITUDE: 275  FLAG: 0 FLAG: 0 STEP: 4
XDATA: -985     YDATA: 0        ZDATA: 128      MAGNITUDE: 993  FLAG: 0 FLAG: 0 STEP: 4
XDATA: -985     YDATA: 64       ZDATA: -32320   MAGNITUDE: 32335        FLAG: 1 FLAG: 1 STEP: 4
XDATA: 32       YDATA: 320      ZDATA: 0        MAGNITUDE: 321  FLAG: 1 FLAG: 0 STEP: 5
XDATA: 32       YDATA: 64       ZDATA: -32064   MAGNITUDE: 32064        FLAG: 1 FLAG: 1 STEP: 5
XDATA: 32       YDATA: 64       ZDATA: -32320   MAGNITUDE: 32320        FLAG: 1 FLAG: 1 STEP: 5
XDATA: -991     YDATA: 64       ZDATA: -31808   MAGNITUDE: 31823        FLAG: 1 FLAG: 1 STEP: 5
XDATA: -985     YDATA: 256      ZDATA: -32512   MAGNITUDE: 32527        FLAG: 1 FLAG: 1 STEP: 5
XDATA: -975     YDATA: 320      ZDATA: 128      MAGNITUDE: 1034 FLAG: 1 FLAG: 0 STEP: 6
```

Figure 22: Algorithm counting steps

With this algorithm, the accelerometer counts a few extra steps because sometimes the threshold value is not crossed on the negative slope—right after a “step” is over. This results in the flag being raised but not lowered. As the number of steps increases, the values smooth out and steps taken evens out with steps counted.

It is worth noting, that the data with the most influence in MAGNITUDE within the algorithm is ZDATA because of the orientation of the chip on the board (for prototyping). For the final delivery, the pedometer must be calibrated again due to the change in orientation. The x and y axes will have a greater influence in data spikes and the threshold will be slightly different but easily determined when the armband is worn.

The algorithm used to code the Arduino is copied below.

```
//SPI library
#include <Arduino.h>
#include <ADXL362.h>
#include <SPI.h>

ADXL362 xl;

// Sensor data variables per axis
int16_t x_data; // 16 bits for x-axis
int16_t y_data; // 16 bits for y-axis
int16_t z_data; // 16 bits for z-axis
int16_t t_data; // 16 bits for temperature data

unsigned long magnitude; // square root of all three axes
unsigned long square_x,square_y, square_z; // squared values
unsigned long add_all; // added, squared values
unsigned long threshold = 1400; // walking threshold
byte flag = 0; // flag for passing threshold
int count = 0; // step counter

void setup()
{
  Serial.begin(9600);
  xl.begin(10);    // Set SPI, soft reset
  xl.beginMeasure();  // Switches on sensor's "measure mode"
  Serial.println("Start of data read");
}

void loop()
{
  int16_t ChipSelect = 10;
  int16_t SlaveSelect = ChipSelect;

  // Gather data
  digitalWrite(SlaveSelect, LOW); // Chip select, starts transmission
  SPI.transfer(0x0B);  // Read register address
  SPI.transfer(0x0E);  // x-axis data register (highest priority)

  x_data = SPI.transfer(0x00);
  x_data = x_data + (SPI.transfer(0x00) << 8);

  y_data = SPI.transfer(0x00);
  y_data = y_data + (SPI.transfer(0x00) << 8);

  z_data = SPI.transfer(0x00);
  z_data = z_data + (SPI.transfer(0x00) << 8);

  t_data = SPI.transfer(0x00);
```

```
t_data = t_data + (SPI.transfer(0x00) << 8);

digitalWrite(SlaveSelect, HIGH); // Chip select, ends transmission

// Print data in monitor
Serial.print("\nXDATA: ");
Serial.print(x_data);
Serial.print("\tYDATA: ");
Serial.print(y_data);
Serial.print("\tZDATA: ");
Serial.print(z_data);
//Serial.print("\tTDATA: ");
//Serial.print(t_data);

square_x = pow(x_data, 2);
square_y = pow(y_data, 2);
square_z = pow(z_data, 2);

add_all = square_x + square_y + square_z;

magnitude = sqrt(add_all);
Serial.print("\tMAGNITUDE: ");
Serial.print(magnitude);

if (magnitude > threshold)
{
  flag = 1;
}

Serial.print("\tFLAG: ");
Serial.print(flag);

if (magnitude < threshold and flag == 1)
{
    count++;
    flag = 0;
}

Serial.print("\tFLAG: ");
Serial.print(flag);
Serial.print("\tSTEP: ");
Serial.print(count);

delay(1000); // Read data easily
```

Thermometer / Accelerometer Tests

While testing the thermometer, the team connected the output voltage of the thermometer (V_{DD}) to an I/O pin of the microcontroller. In order to test that the thermometer worked as expected, the team gather data from various temperature sources and compared these values with those from a digital thermometer available to students in the Senior Design lab. As shown in the Design and Evidence section of this subsystem, the tested values were very close to the expected temperature values.

While testing the accelerometer, an Arduino microcontroller was used to understand SPI protocol and create a step counting algorithm. The resulting algorithm and test results is shown in the Design and Evidence section of this subsystem as well. As with the thermometer, the values gathered from testing are similar to the expected values, with the slight note that in order for the data to be more accurate, more steps should be taken.

Thermometer / Accelerometer Prototyping

Although the components of this subsystem were not soldered onto the printed PCB, they were placed in a breadboard in order to demonstrate that the whole system of the armband was able to interact as a whole as pictured below:

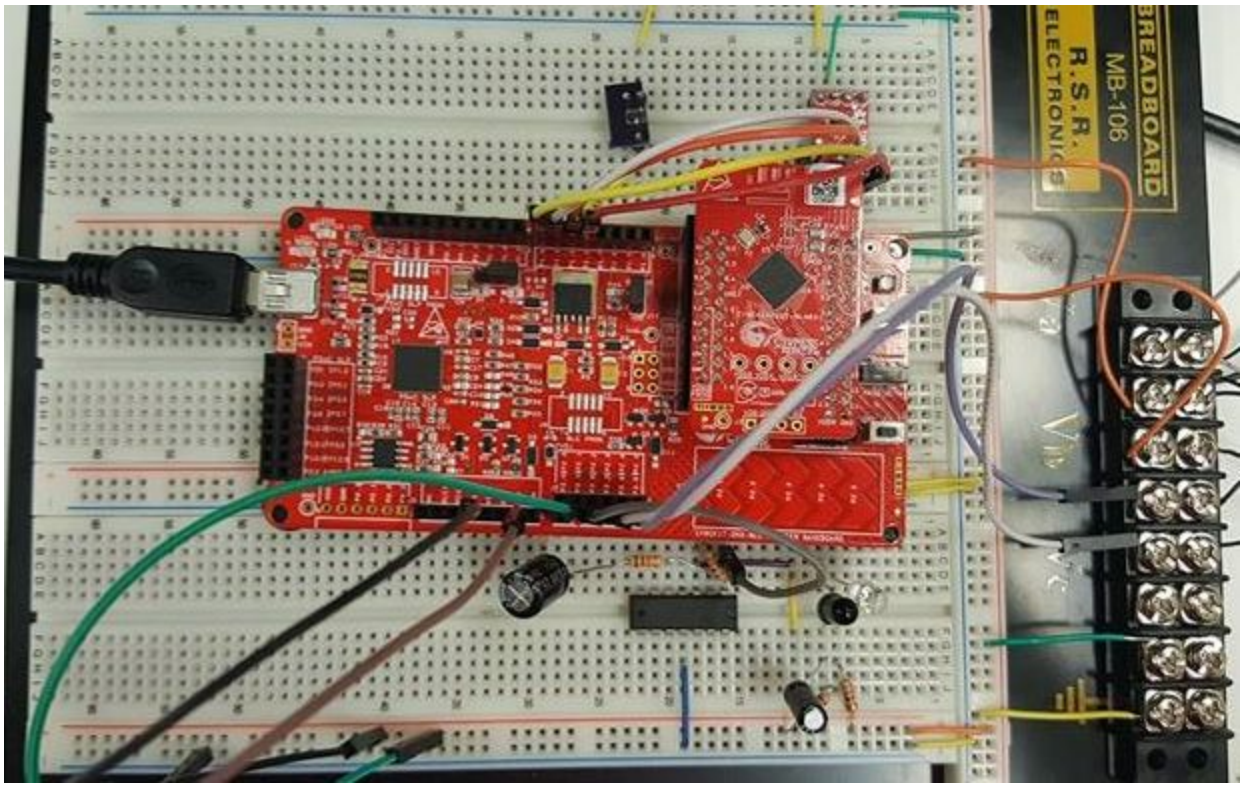


Figure 23: Thermometer / accelerometer prototype testing

These components were not placed on the printed PCB because the team was not able to program the microcontroller on the PCB, and the components would not have worked unless there was communication with the microcontroller. Therefore, the team decided that the best way to demonstrate the state of the whole system was to use the prototype microcontroller and use is as a power source as well (we were not able to program the microcontroller while powering the components independently from the breadboard or the rechargeable battery subsystem).

Technical overview of Heart Rate subsystem

Heart rate sensor consists of an infrared light emitting diode and a photodiode. Due to variation in the flow of blood to various regions of human body, heart beat pulses are generated. When a finger tissue is illuminated with light source, the IR LED transmits an infrared light into the fingertip, a part of which is reflected from the blood inside of arteries of finger. The photo transistor detects the portion of the light which is reflected back. Whenever the heart beats the amount of reflected infrared light changes, it is able to be perceived by the phototransistor.

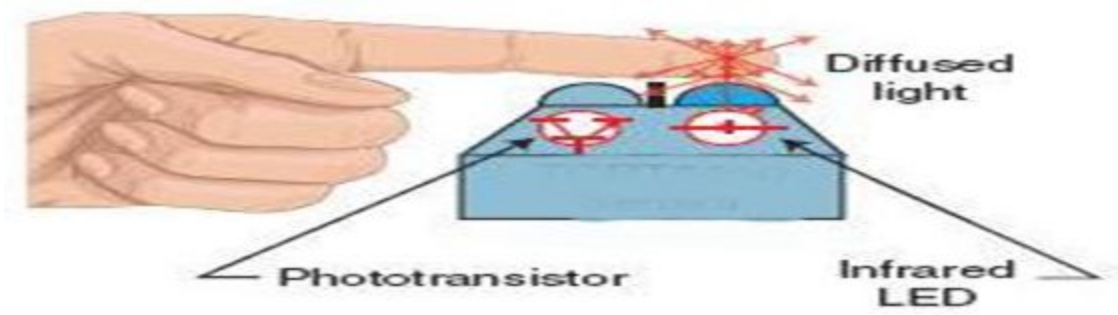


Figure Heart Rate Sensor
http://fos.cmb.ac.lk/esl/wp-content/uploads/2013/05/a.jpg

Detailed subsystem block diagram

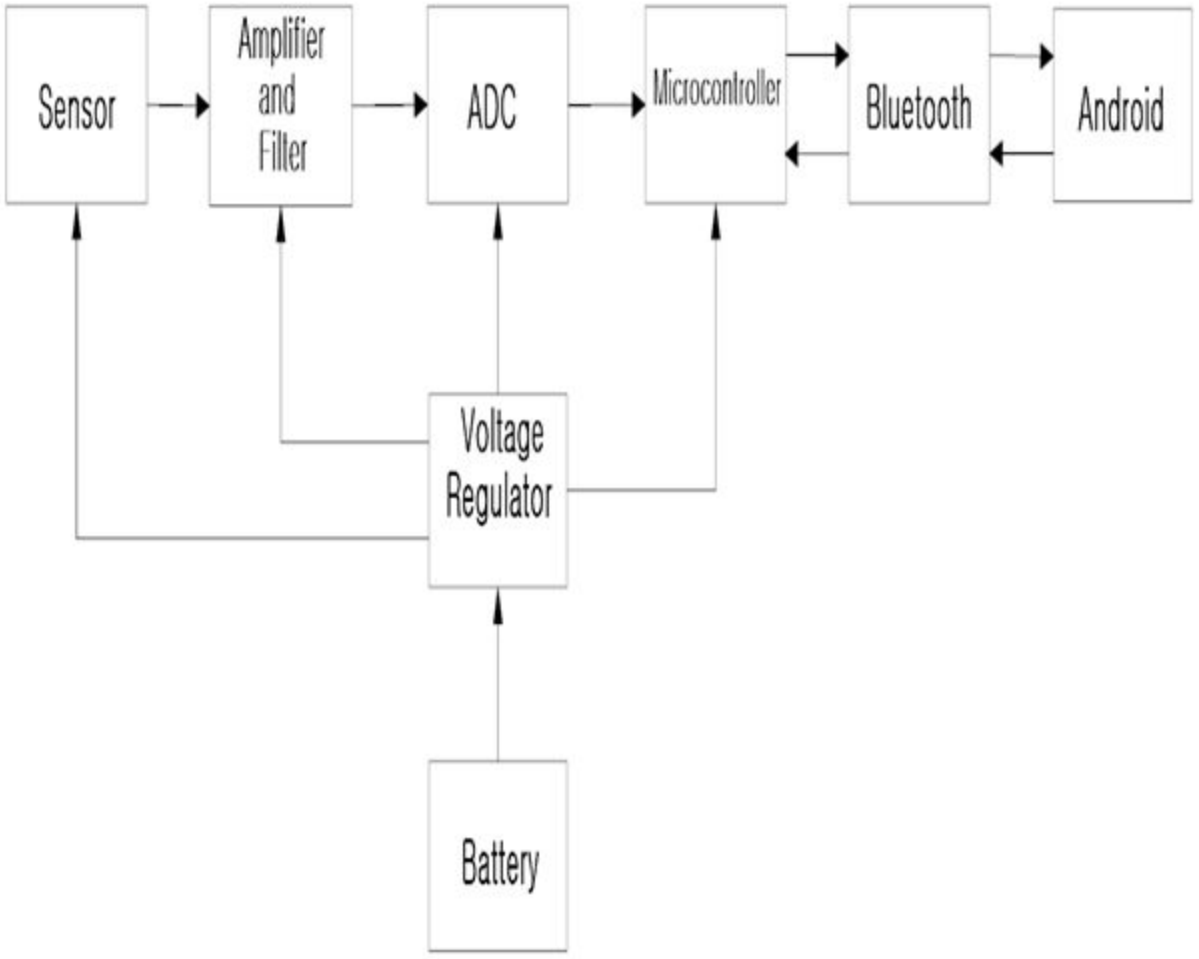


Figure Block diagram of Heart Rate Monitor

List of subsystem requirements

Infrared Transmitter LED

Infrared transmitter light emitting diode is known as IR transmitter which transmit infrared rays in 940nm range for this project

Figure infrared Transmitted LED
http://www.vishay.com/docs/81300/vsml3710.pdf

Phototransistor

Photo-transistor is a device which converts light energy into electric energy. This transistor is encased in clear container to enhance light as it travels through it and allow the light to reach the its sensitive parts. As the current draws from base to the emitter, the current would be converted to voltage.

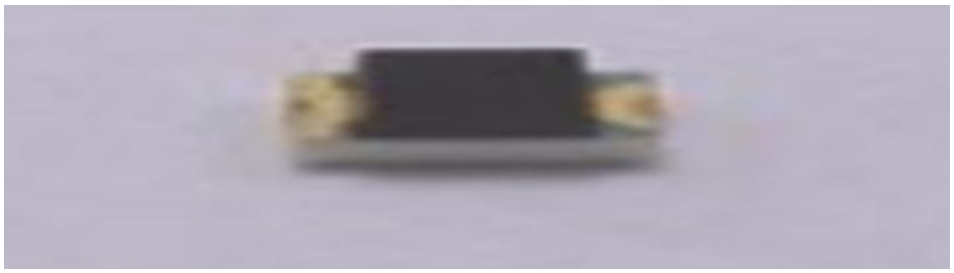


Figure Phototransistor
http://www.everlight.com/file/ProductFile/PT17-21B-L41-TR8.pdf

Operational Amplifier (LM324)

An operational amplifier is a voltage amplifying device which is designed to be used with external feedback components such as resistors and capacitors between its output and input terminals. The feedback of electrical components would determine the result of function.

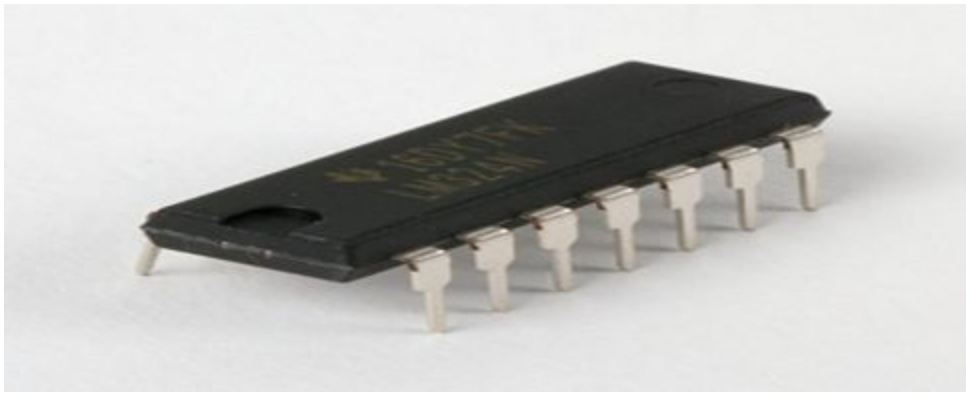


Figure LM324
http://www.ashopbd.com/wp-content/uploads/2016/04/LM324.jpg

Description of Heart Rate Sensor with Cypress Microcontroller

The power would be provided as 5V from voltage regulator which makes Heart Rate sensor works properly. The Heart Rate Sensor also is connected to Microcontroller which would read ADC from the circuit. Heart rate signal circuit consists of high pass filter with a cut off frequency of about 1.59Hz and value of gain would become 11. The stage of amplifier provides sufficient gain to boost weak signal from phototransistor and it would be converted into in pulse. Through the progress of this, heart beat is detected and the output form the signal goes to input of the microcontroller. A value of DC offset became 2.5V from 5V which is half value of VCC. To obtain 2.5V, the team connected 220kohm in series that led VCC / 2 and also block higher frequency noises present in the signal as put 1uf and 100uf capacitor. Therefore, the signal is able to be read by the microcontroller.

Evidence

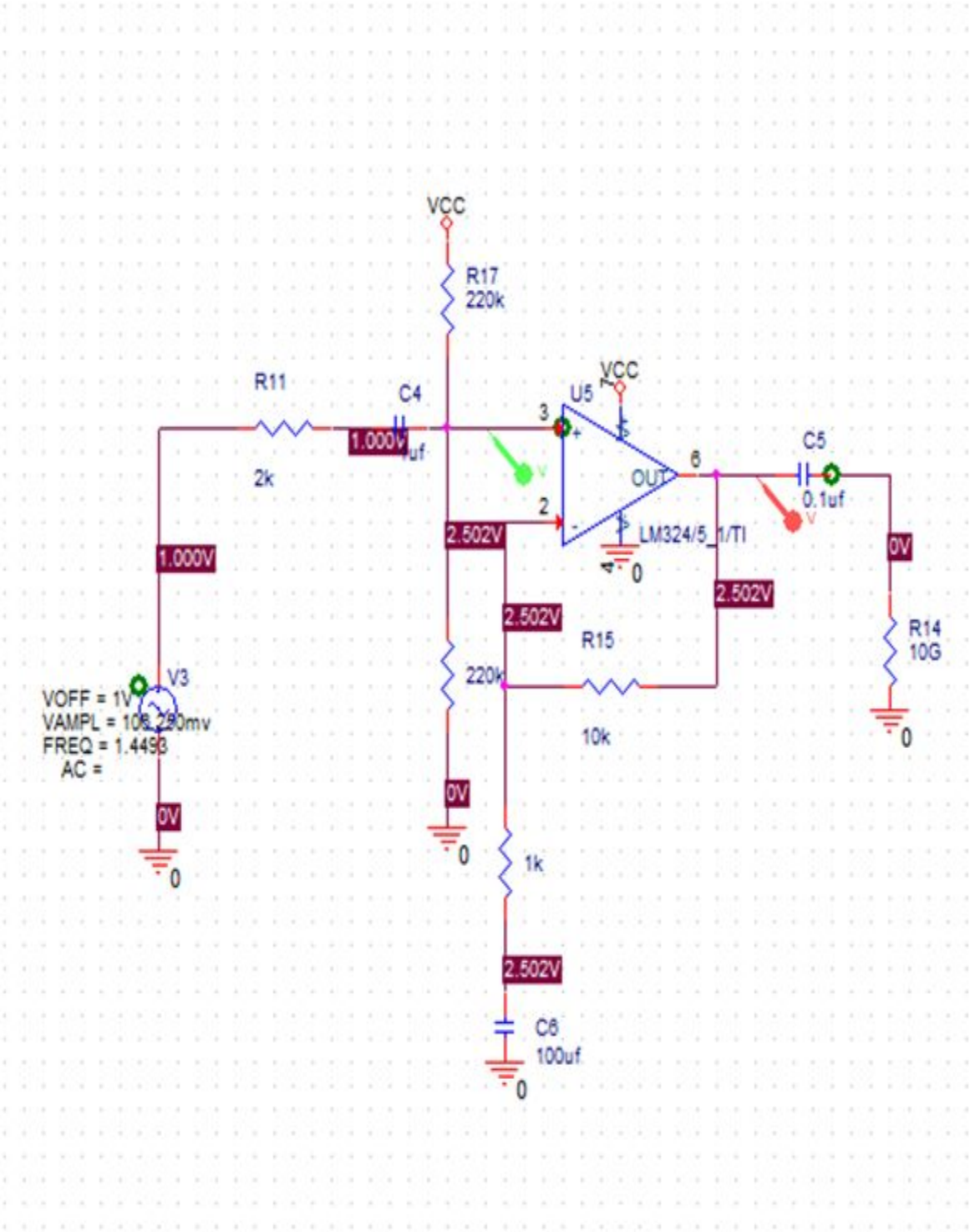

VCC
R17
220k
R11
2k
C4
1.000V
uf
VCC
U5
3
2
6
OUT
LM324/5_1/TI
4
0
C5
0.1uf
2.502V
1.000V
2.502V
2.502V
R15
220k
10k
R14
10G
0V
V3
VOFF = 1V
VAMPL = 100.280mv
FREQ = 1.4493
AC =
0V
0
0V
0
1k
2.502V
C6
100uf
0

Figure Simulation of Heart Rate Sensor

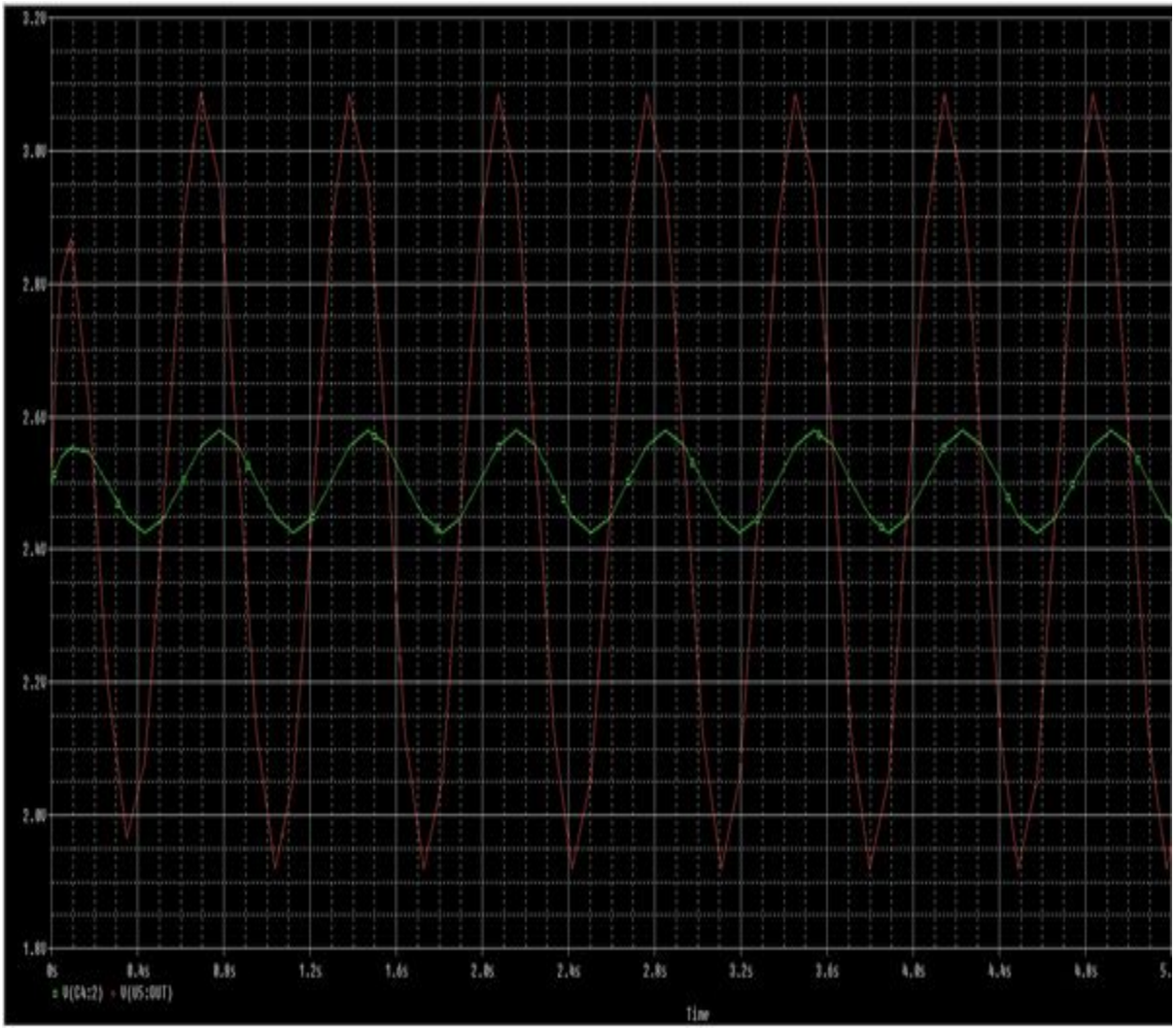


Figure Result of Simulation

To figure out if the circuit works, the team decided to use ORCAD software to simulate with the circuit. As the team measured voltage differences between input and of operational amplifier, a value of DC offset became 2.5V from 5V which is half value of VCC. To obtain 2.5V, the team connected 220k ohm in series that led Vcc / 2. With the result of this, the AC signal can be read by microcontroller.

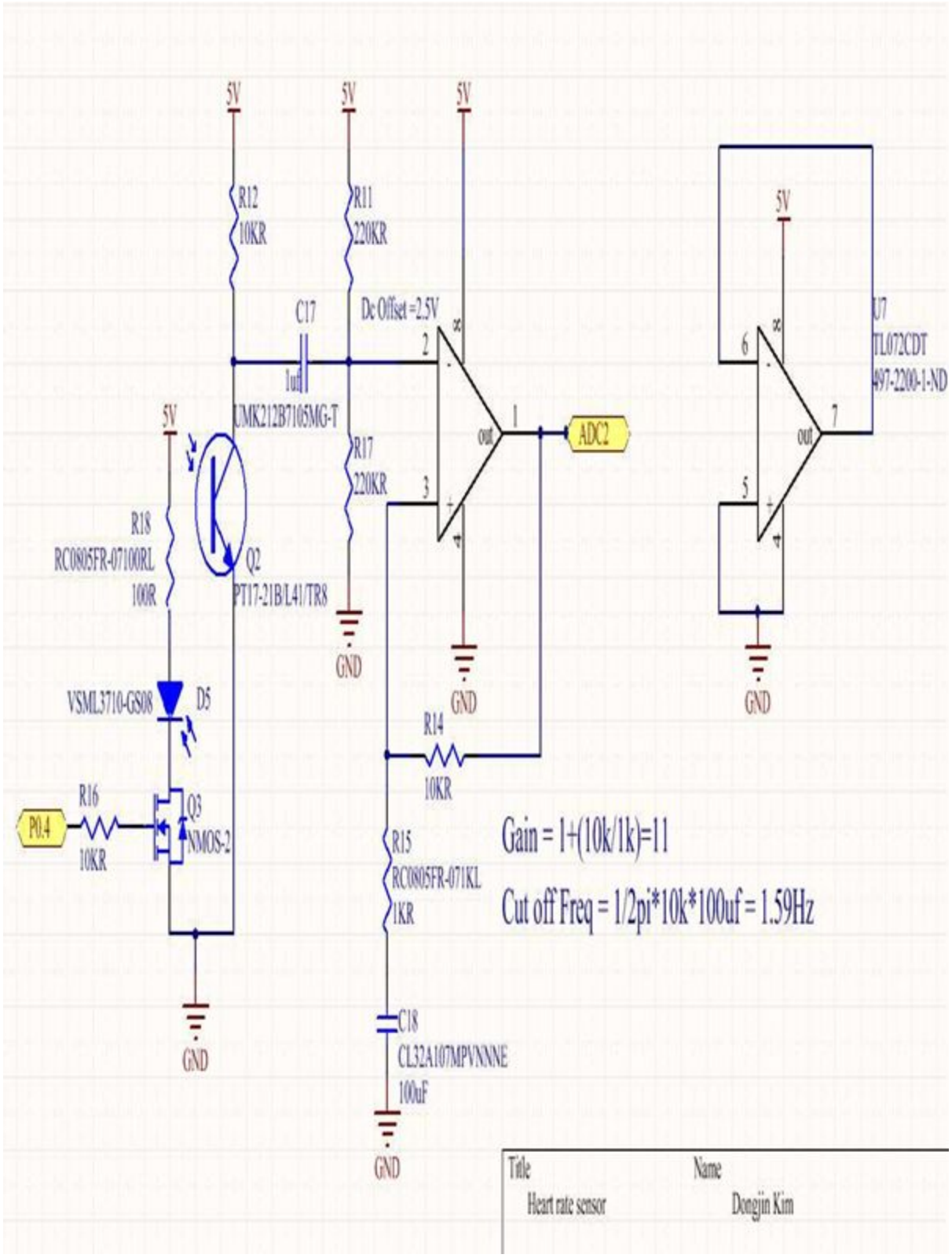


Figure Schematic of Heart Rate Sensor

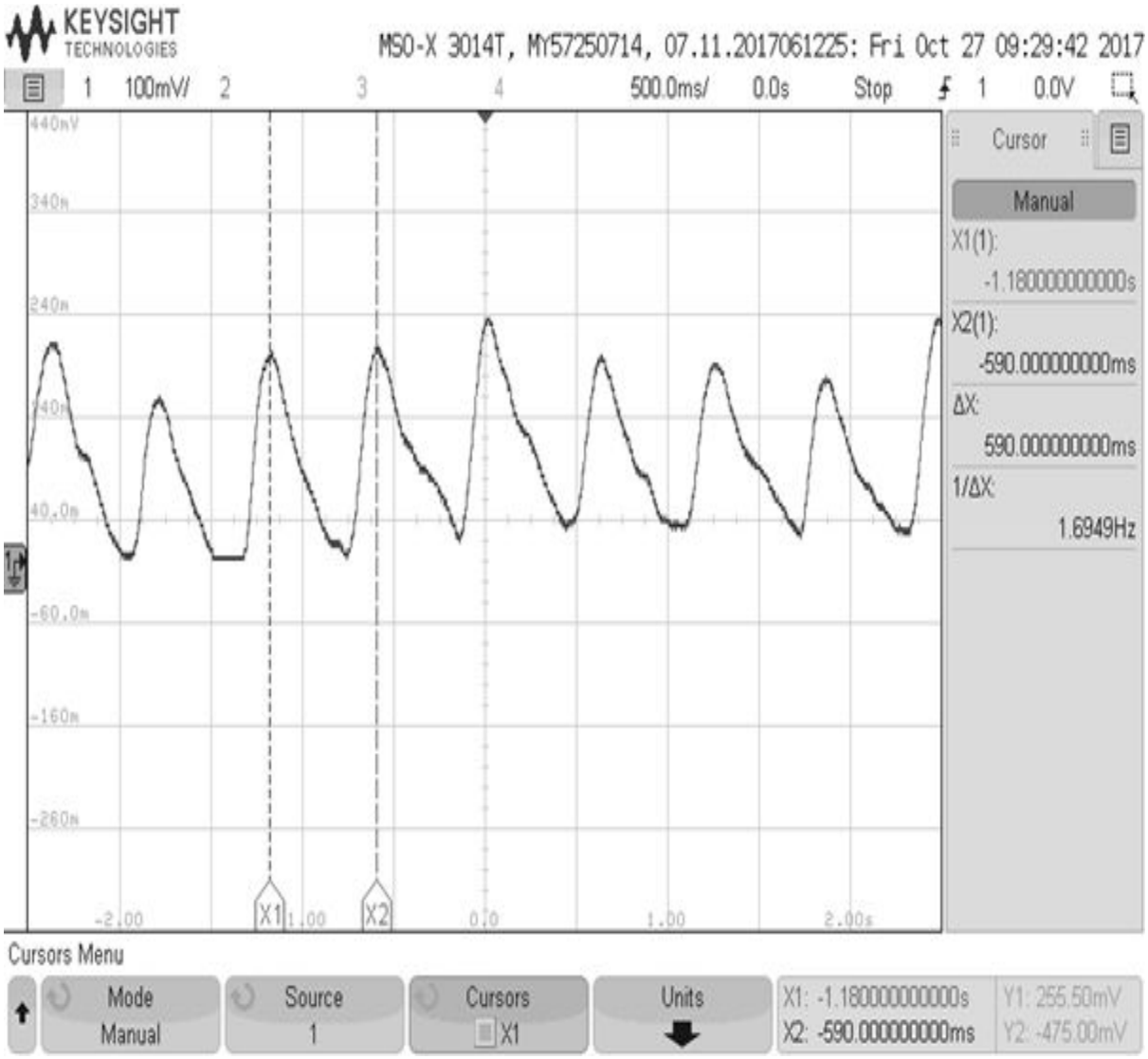


Figure Signal of Heart Rate Sensor

When our team finished with connection, the signal of heart rate showed up in oscilloscope that measured as one of members touched the phototransistor and IR LED.

Equation of conversion from Hz to BPM

Hertz = BPM / 60, BPM = Hertz * 60

1.6949 * 60 = 101.694 BPM

Tests

Equipment List
-Power Supply
-Oscilloscope
-Circuit of Heart Rate Sensor

Connection Diagram

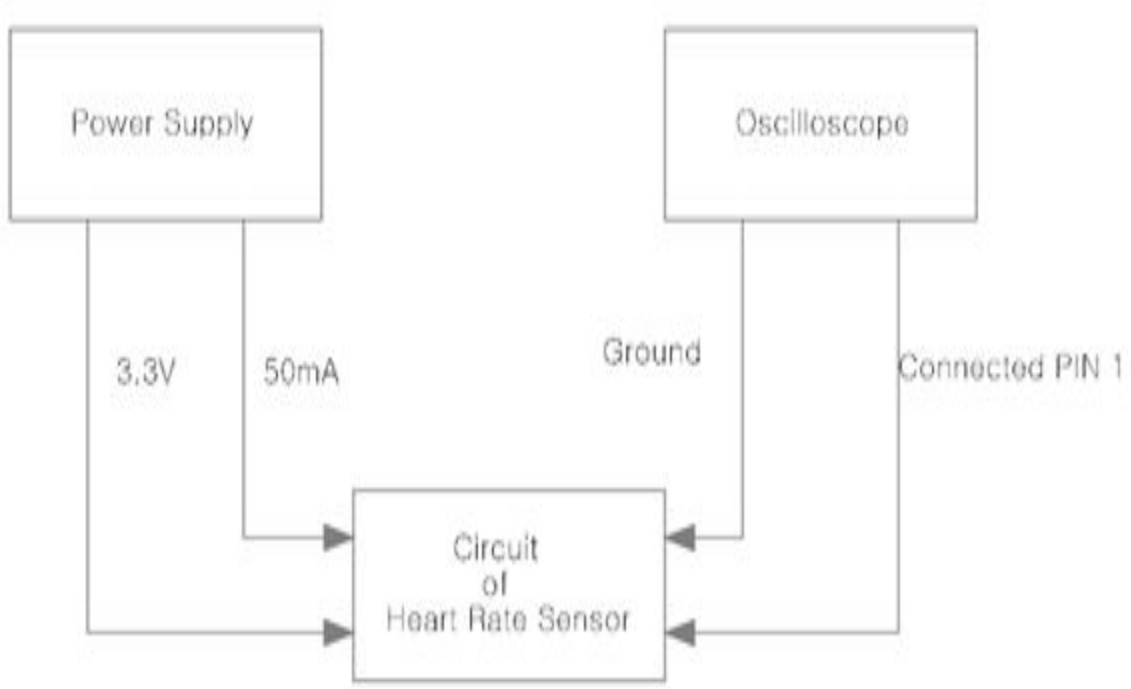


Figure Connection diagram of Heart Rate

To test out with hardware part, the team connected power supply to the circuit with value of 3.3V and 50mA and measured with oscilloscope as connected to the circuit also. As a result of the test, it showed up with 1.6949 Hz which is also same as 101.694 BPM.
Equation of conversion from Hz to BPM
Hertz = BPM / 60, BPM = Hertz * 60
1.6949 * 60 = 101.69**4**

Instructions
After finish connection among power supply, the circuit, and oscilloscope, one of members put his finger between IR LED and phototransistor and it brings the result.

Photo of Prototype

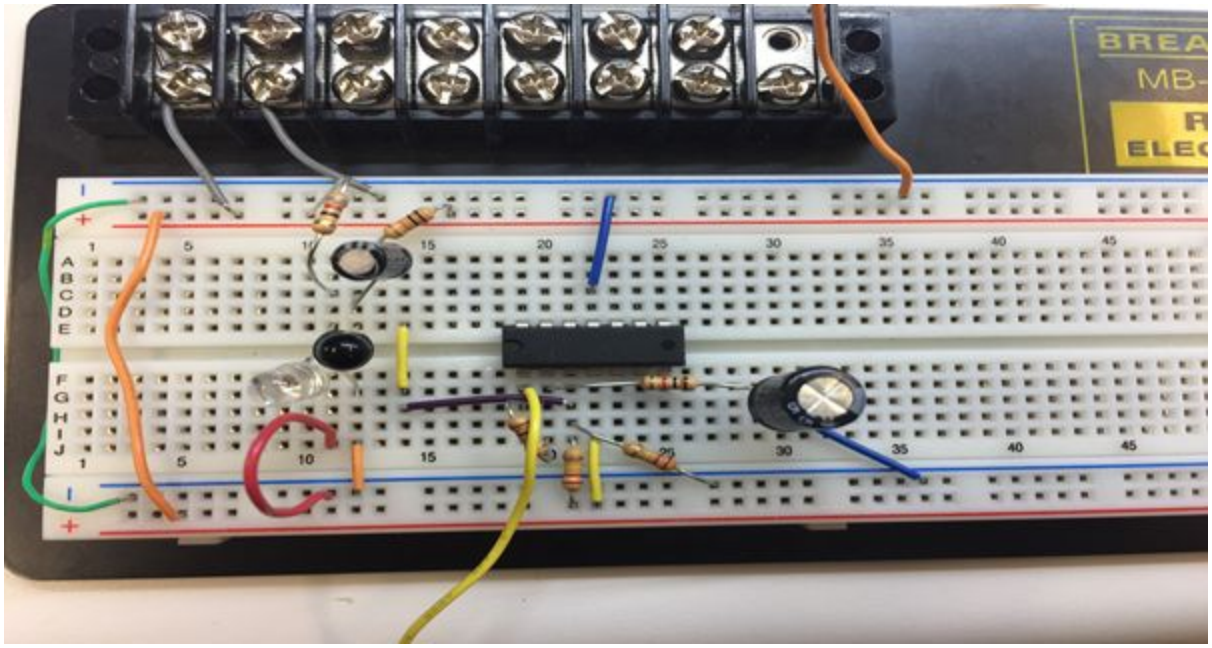


Figure Prototype of only Heart Rate Sensor

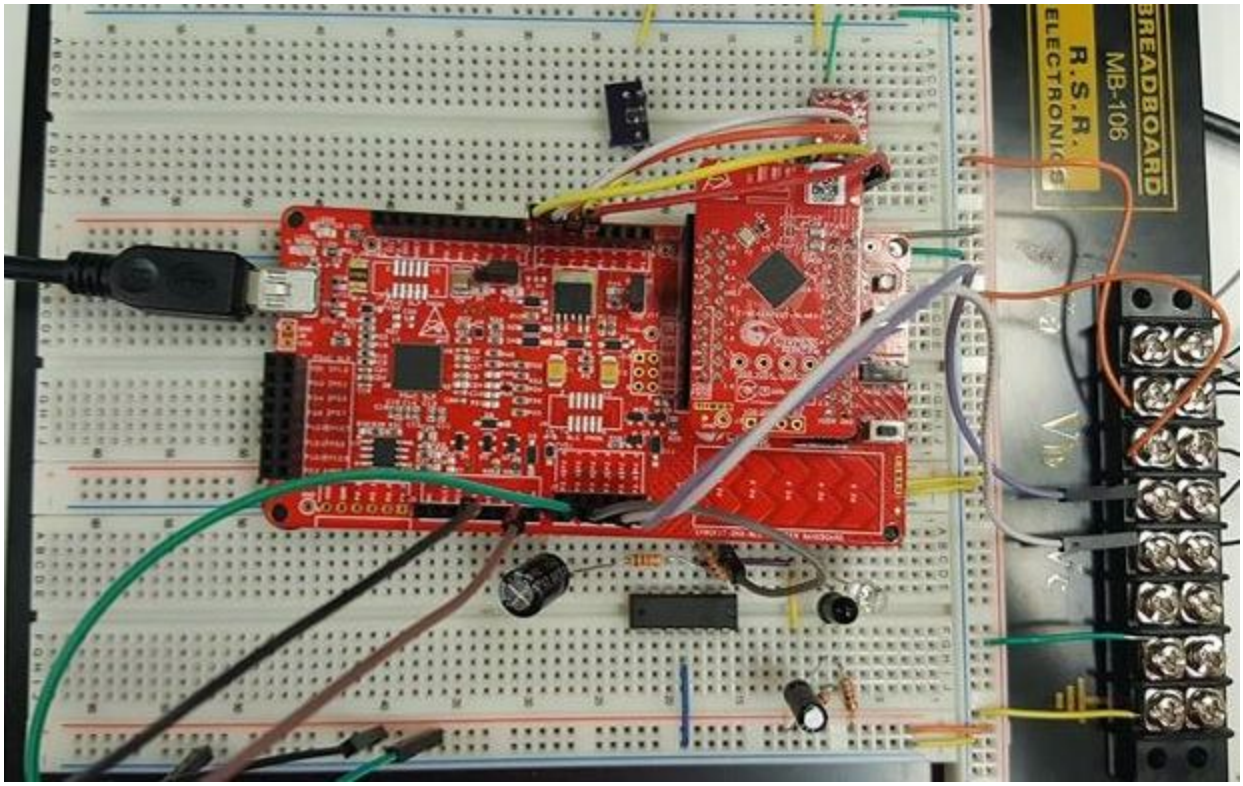


Figure Prototype of assembled device

Technical overview of GPS

GPS is navigation system based on satellite. The system made up of at least of 24 satellites and also works anywhere in the world. The satellites circle the Earth twice a day in a precise orbit. Every single satellite transmits a certain signal and parameters that let GPS device to calculate the precise location of the satellite. Therefore, GPS receivers use this information to compute users' location.

Detailed subsystem block diagram

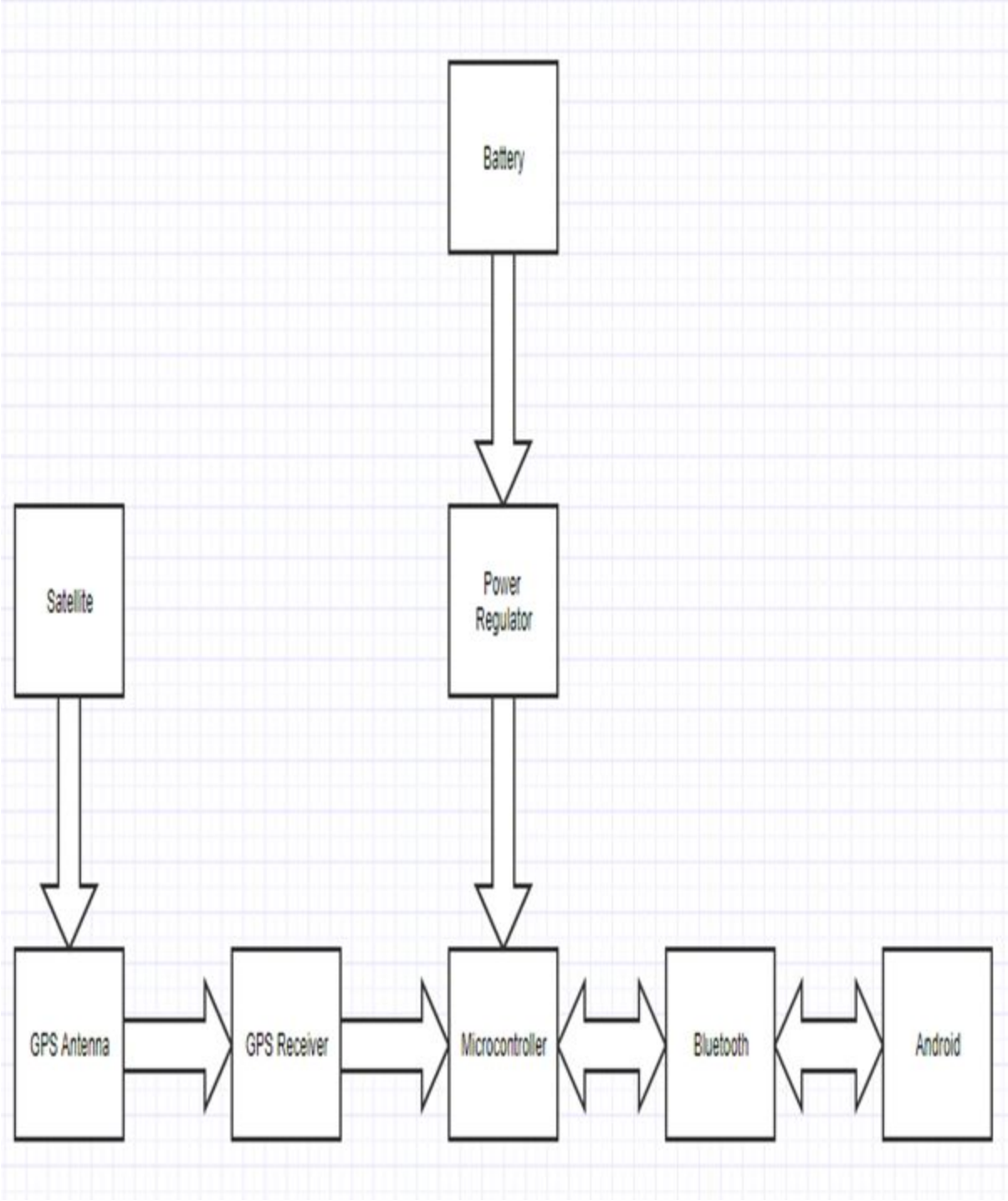


Figure GPS Block diagram

List of subsystem requirement with explanation

Equipment

- GPS 735 Module

Figure GPS Module

https://cdn.sparkfun.com/datasheets/GPS/GP-735T-150203.pdf

The GP 735 which is one of GPS modules that easy to use antenna receiver. There are 56 channel GPS module which based on the U-Blox that has operating voltage from 3.3v to 5v. There is also power saving mode which would be worked by microcontroller that would help saving power. However, our team decided not to use power saving mode because it is not necessary for our design project.

- U-blox software

This is evaluation software for GPS module which provides current location when a circuit of GPS module connected to PC.

GPS module interfacing with Cypress Microcontroller

3.3V and would be charged for GPS module and microcontroller to be worked by voltage regulator. The GPS module always transmits serial data in form of sentences. Through this, values of longitude and latitude of a certain location would be contained as the sentences. TX which is serial data output from GPS, Rx which is serial data input to GPS and GND(ground) would be needed to communicate over UART with the microcontroller. In order to figure out a certain location of the GPS receiver based on longitude and latitude, the module would provide output data in logic level format. Using the logic level format, time, date, location would be determined such as Figure Description of GPS.

Evidence

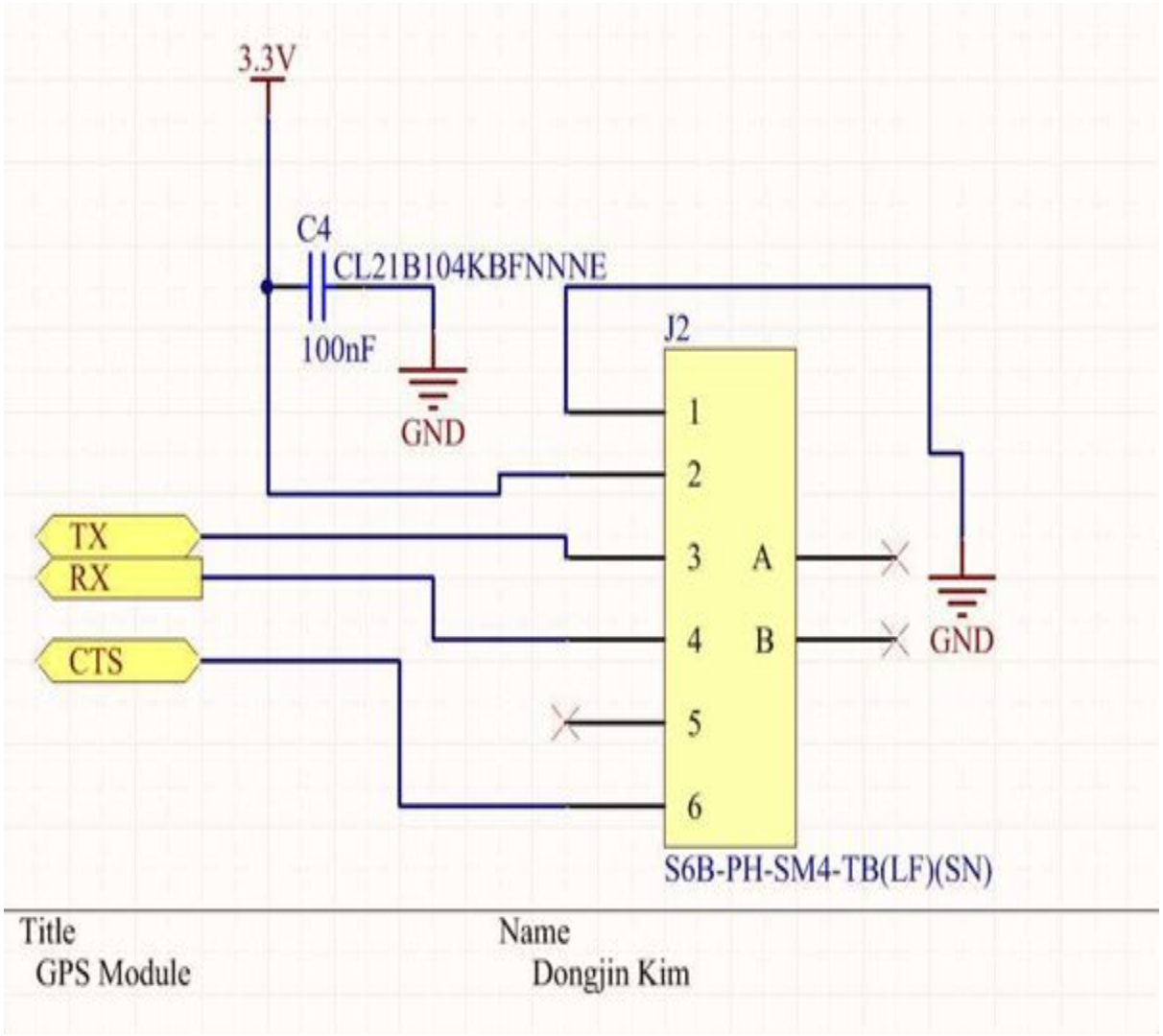


Figure Schematic of GPS

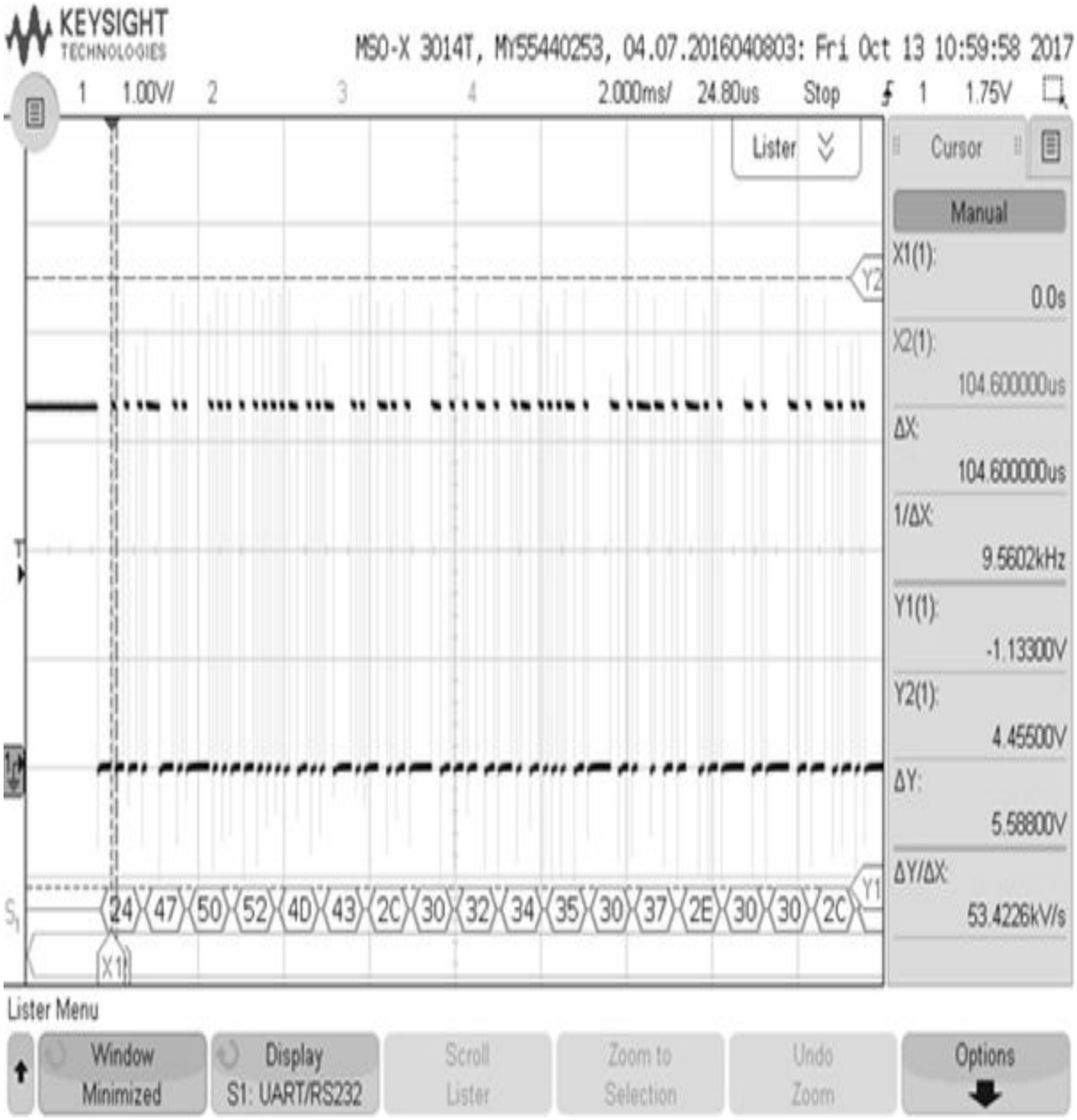


Figure Data of GPS using oscilloscope

As the team connected GPS module to power supply and oscilloscope, it measured as ‘Figure Data of GPS using oscilloscope’. Through this, the data was determined as user could see above the figure. The team obtained numbers bottom part of the oscilloscope. The numbers were 24, 47, 50, 52, 4D, and 43 then translated to word that showed as ‘$GPRMC’ which is known as RMC protocol header.

Contents	Example	Unit	Explanation
Message ID	$GPRMC		RMC protocol header
UTC Time	065500.00		hhmmss.ss hh: hour, mm: minute, ss: second
Status	A		A: Data valid, V: Data invalid
Latitude	2447.65027		ddmm.mmmmm dd: degree, mm.mmmmm: minute
North/South	N		N: North Latitude, S: South Latitude
Longitude	12100.78318		dddmm.mmmmm dd: degree, mm.mmmmm: minute
East/West	E		E: East Longitude, W: West Longitude
Speed over ground	15.869	knots	Receiver's speed
Course over ground	189.32	degrees	Receiver's direction of travel Moving clockwise starting at due north
Date	051109		ddmmyy dd: Day, mm: Month, yy: Year
Magnetic variation		degrees	This receiver does not support magnetic declination. All "course over ground" data are geodetic WGS84 directions.
Mode Indicator	D		A: Autonomous M: Manual D: DGPS S: Simulation E: Dead Reckoning N: Data Invalid
checksum	*57		
<CR><LF>			End of sentence

Fig. Description of GPS

https://cdn.sparkfun.com/datasheets/GPS/GP-735T-150203.pdf

The figure (Description of GPS) above shows that how to read UTC Time, status, latitude, longitude and date. For example, if the number shows as 065500.00 that would could be translated as hhmmss.ss and also may be analyzed hh:hour, mm:minute, ss:second. For Latitude (2447.65027), it would be read as degree and minute and dd:degree, mm,mmmmm: minute. At the last, if 051109 shows up and then would be read as day month and year.

Tests : equipment list, connection diagram, and instruction

Equipment list

- Power Supply : Input for voltage 3.3V, input for current 50mA
- Oscilloscope : AC and high revolution mode
- U-block software : Current location would be determined by this software as connected to the GPS module and PC.

Connection Diagram

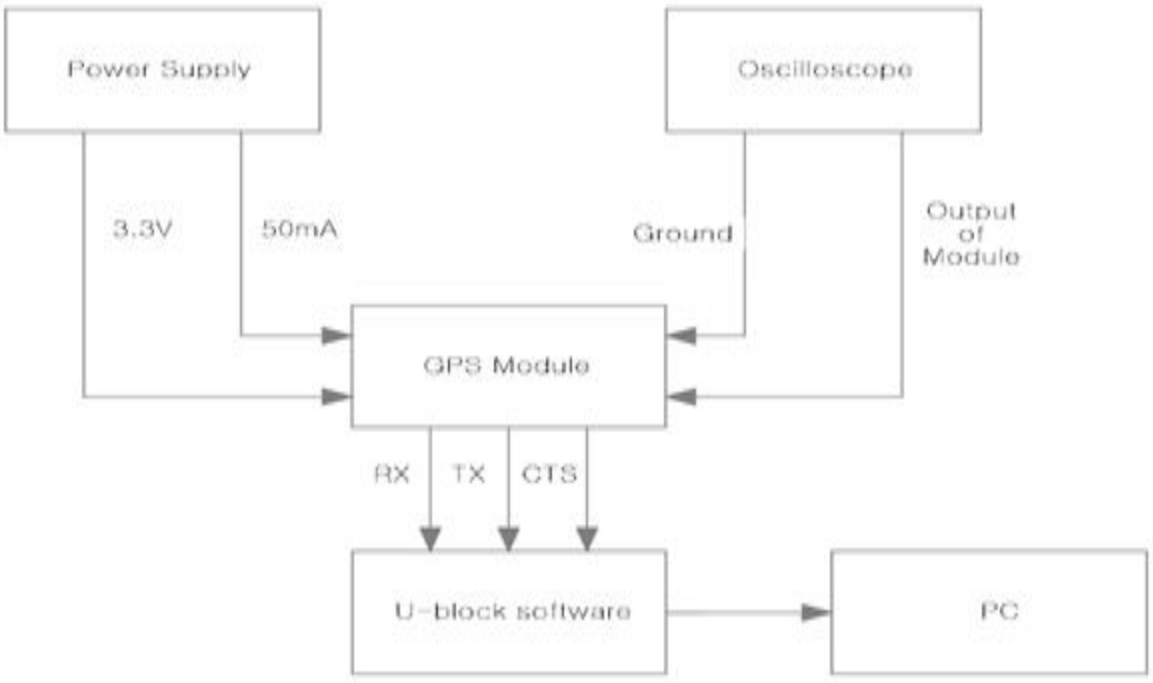


Figure Connection of Diagram

Instruction

As the team connected GPS module to power supply and oscilloscope, it measured as ‘Figure Data of GPS using oscilloscope’. Through this, the data would be determined as user could see below the figure

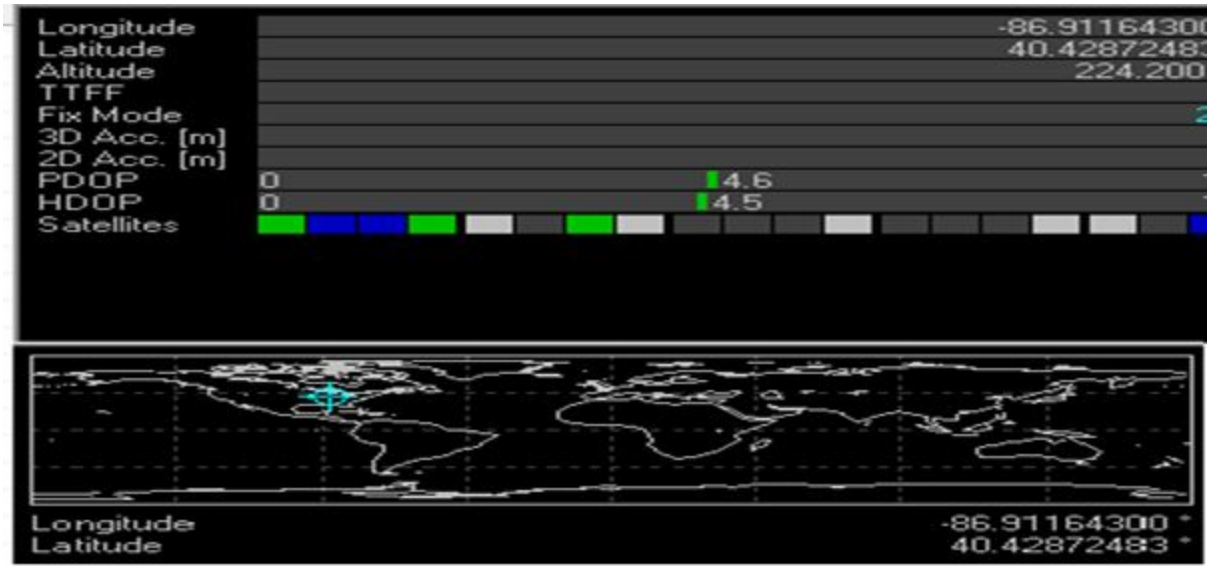


Figure Result of Test for current location

Photo of prototype

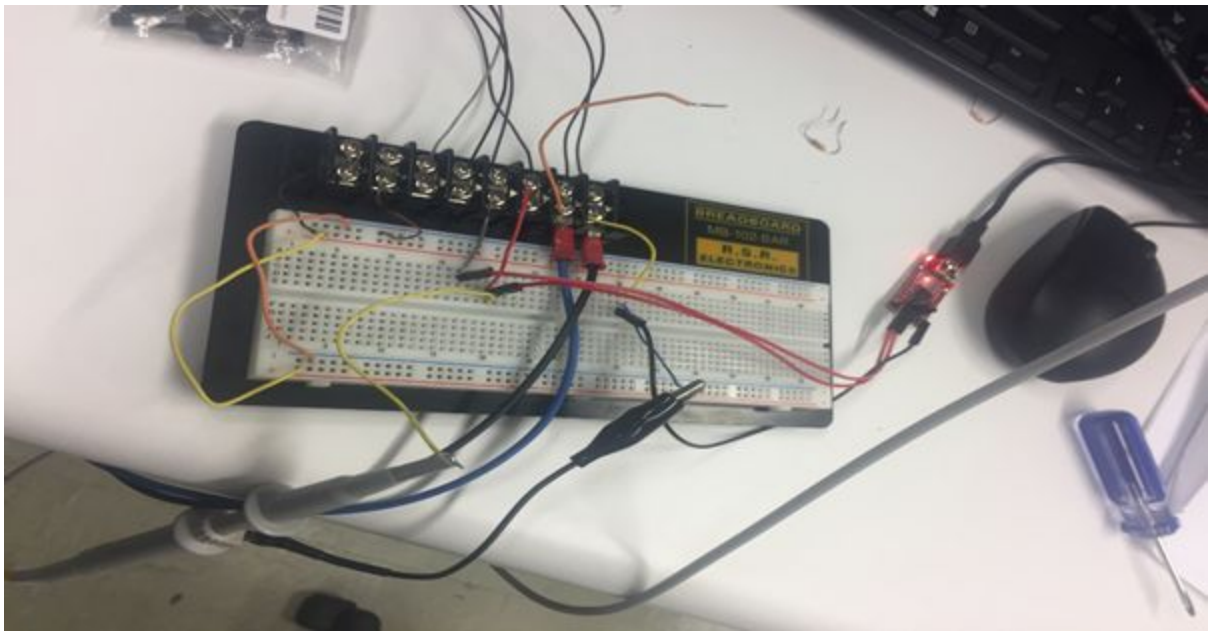


Figure Photo of prototype

Appendice

PKCELL

Li-Polymer Battery Technology Specification

Model： Li-Polymer 503035 500mAh 3.7V with PCM

Customer confirmation	Corporate name	
	Checked	
	Approved	
	Corporate seal	

Signed：_____

Drafted by：_____

Approved by：_____

Document No.： **QA.S.0228** Edit：**A**

SHENZHEN PKCELL BATTERY CO., LTD
Company address: E2 Building,Guangming Technology Park,No.24 Zhonghua Road,Longhua New Area,Shenzhen Guangdong, China.
（Tel）：+86-755-86670672 （Fax）：+86-755-86670609
E-mail：pkcell@pkcell.net Website：http://www.pkcell.net

CY8CKIT-145-40XX

PSoC® 4000S Prototyping Kit Guide

Doc. # 002-11504 Rev. *A

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

National Semiconductor

December 1995

TL081 Wide Bandwidth JFET Input Operational Amplifier

General Description

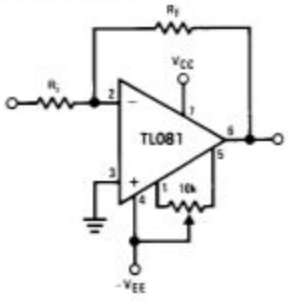
The TL081 is a low cost high speed JFET input operational amplifier with an internally trimmed input offset voltage (BI-FET II™ technology). The device requires a low supply current and yet maintains a large gain bandwidth product and a fast slew rate. In addition, well matched high voltage JFET input devices provide very low input bias and offset currents. The TL081 is pin compatible with the standard LM741 and uses the same offset voltage adjustment circuitry. This feature allows designers to immediately upgrade the overall performance of existing LM741 designs.

The TL081 may be used in applications such as high speed integrators, fast D/A converters, sample-and-hold circuits and many other circuits requiring low input offset voltage, low input bias current, high input impedance, high slew rate and wide bandwidth. The devices has low noise and offset voltage drift, but for applications where these requirements are critical, the LF356 is recommended. If maximum supply current is important, however, the TL081C is the better choice.

Features

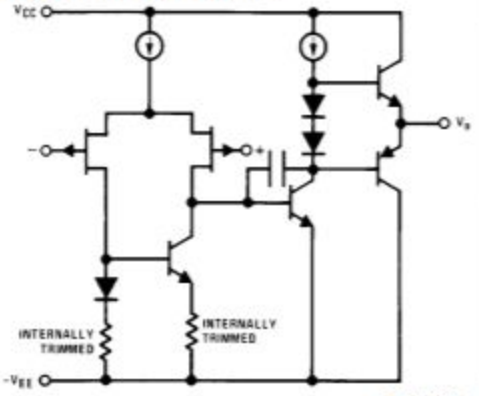
■ Internally trimmed offset voltage	15 mV
■ Low input bias current	50 pA
■ Low input noise voltage	25 nV/$\sqrt{Hz}$
■ Low input noise current	0.01 pA/$\sqrt{Hz}$
■ Wide gain bandwidth	4 MHz
■ High slew rate	13 V/μs
■ Low supply current	1.8 mA
■ High input impedance	$10^{12}\Omega$
■ Low total harmonic distortion A_V = 10, R_L = 10k, V_O = 20 Vp-p, BW = 20 Hz–20 kHz	<0.02%
■ Low 1/f noise corner	50 Hz
■ Fast settling time to 0.01%	2 μs

Typical Connection



TL/H/8358–1

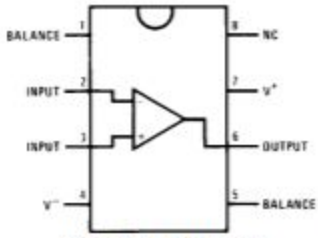
Simplified Schematic



TL/H/8358–2

Connection Diagram

Dual-In-Line Package



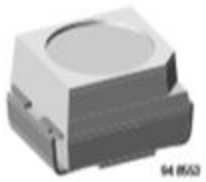
TL/H/8358–4

Order Number TL081CP
See NS Package Number N08E

BI-FET II™ is a trademark of National Semiconductor Corp.

RRD-B30M125/Printed in U. S. A.

High Power Infrared Emitting Diode, RoHS Compliant, 940 nm, GaAlAs/GaAs



FEATURES

- Package type: surface mount
- Package form: PLCC-2
- Dimensions (L x W x H in mm): 3.5 x 2.8 x 1.75
- Peak wavelength: λ_p = 940 nm
- High reliability
- High radiant power
- High radiant intensity
- Angle of half intensity: $\varphi = \pm 60°$
- Low forward voltage
- Suitable for high pulse current operation
- Good spectral matching with Si photodetectors
- Package matched with IR emitter series VEMT3700
- Floor life: 4 weeks, MSL 2a, acc. J-STD-020
- Lead (Pb)-free reflow soldering
- Lead (Pb)-free component in accordance with RoHS 2002/95/EC and WEEE 2002/96/EC

RoHS COMPLIANT

DESCRIPTION

VSML3710 is an infrared, 940 nm emitting diode in GaAlAs/GaAs technology with high radiant power, molded in a PLCC-2 package for surface mounting (SMD).

APPLICATIONS

- IR emitter in photointerrupters, sensors and reflective sensors
- IR emitter in low space applications
- Household appliance
- Tactile keyboards

PRODUCT SUMMARY

COMPONENT	I_e (mW/sr)	φ (deg)	λ_p (nm)	t_r (ns)
VSML3710	8	± 60	940	800

Note
Test conditions see table "Basic Characteristics"

ORDERING INFORMATION

ORDERING CODE	PACKAGING	REMARKS	PACKAGE FORM
VSML3710-GS08	Tape and reel	MOQ: 7500 pcs, 1500 pcs/reel	PLCC-2
VSML3710-GS18	Tape and reel	MOQ: 8000 pcs, 8000 pcs/reel	PLCC-2

Note
MOQ: minimum order quantity

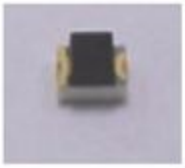
ABSOLUTE MAXIMUM RATINGS

PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
Reverse voltage		V_R	5	V
Forward current		I_F	100	mA
Peak forward current	$t_p/T = 0.5$, $t_p = 100$ µs	I_{FM}	200	mA
Surge forward current	$t_p = 100$ µs	I_{FSM}	1	A
Power dissipation		P_V	160	mW

EVERLIGHT

DATASHEET

0805 Package Phototransistor
PT17-21B/L41/TR8



Features
- Fast response time
- High photo sensitivity
- Small junction capacitance
- Pb free
- The product itself will remain within RoHS compliant version.
- Compliance with EU REACH

Descriptions
- PT17-21B/L41/TR8 is a phototransistor in miniature SMD package which is molded in a black with flat top view lens.
  The device is Spectrally matched to infrared emitting diode.

Applications
- Miniature switch
- Counters and sorter
- Position sensor
- Infrared applied system
- Encoder

Device Selection Guide

Part Category	Chip Material	Lens Color
PT	Silicon	Black

TEXAS INSTRUMENTS

LM124-N, LM224-N
LM2902-N, LM324-N

SNOSC16D – MARCH 2000 – REVISED JANUARY 2015

LMx24-N, LM2902-N Low-Power, Quad-Operational Amplifiers

1 Features

- Internally Frequency Compensated for Unity Gain
- Large DC Voltage Gain 100 dB
- Wide Bandwidth (Unity Gain) 1 MHz (Temperature Compensated)
- Wide Power Supply Range:
  - Single Supply 3 V to 32 V
  - or Dual Supplies ±1.5 V to ±16 V
- Very Low Supply Current Drain (700 µA) —Essentially Independent of Supply Voltage
- Low Input Biasing Current 45 nA (Temperature Compensated)
- Low Input Offset Voltage 2 mV and Offset Current: 5 nA
- Input Common-Mode Voltage Range Includes Ground
- Differential Input Voltage Range Equal to the Power Supply Voltage
- Large Output Voltage Swing 0 V to V⁺ − 1.5 V
- **Advantages:**
  - Eliminates Need for Dual Supplies
  - Four Internally Compensated Op Amps in a Single Package
  - Allows Direct Sensing Near GND and V_{OUT} also Goes to GND
  - Compatible With All Forms of Logic
  - Power Drain Suitable for Battery Operation
  - In the Linear Mode the Input Common-Mode, Voltage Range Includes Ground and the Output Voltage
  - Can Swing to Ground, Even Though Operated from Only a Single Power Supply Voltage
  - Unity Gain Cross Frequency is Temperature Compensated
  - Input Bias Current is Also Temperature Compensated

2 Applications

- Transducer Amplifiers
- DC Gain Blocks
- Conventional Op Amp Circuits

3 Description

The LM124-N series consists of four independent, high-gain, internally frequency compensated operational amplifiers designed to operate from a single power supply over a wide range of voltages. Operation from split-power supplies is also possible and the low-power supply current drain is independent of the magnitude of the power supply voltage.

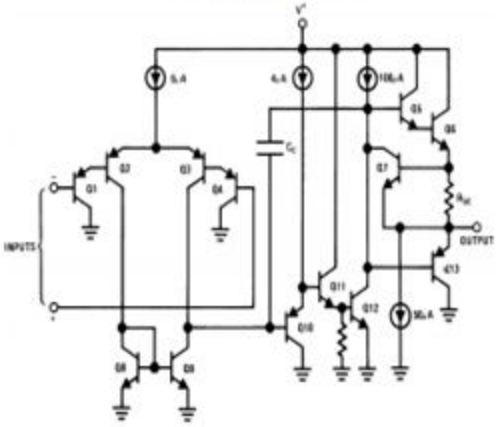
Application areas include transducer amplifiers, DC gain blocks and all the conventional op amp circuits which now can be more easily implemented in single power supply systems. For example, the LM124-N series can directly operate off of the standard 5-V power supply voltage which is used in digital systems and easily provides the required interface electronics without requiring the additional ±15 V power supplies.

Device Information[1]

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM124-N	CDIP (14)	19.56 mm × 6.67 mm
LM224-N		
LM324-N	CDIP (14)	19.56 mm × 6.67 mm
	PDIP (14)	19.177 mm × 6.35 mm
	SOIC (14)	8.65 mm × 3.91 mm
	TSSOP (14)	5.00 mm × 4.40 mm
LM2902-N	PDIP (14)	19.177 mm × 6.35 mm
	SOIC (14)	8.65 mm × 3.91 mm
	TSSOP (14)	5.00 mm × 4.40 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

Schematic Diagram



ADH-tech ADH Technology Co. Ltd.

Data Sheet / GP-735

Easy to Use,

Slim,

Ultra High Performance,

GPS

Smart Antenna Module

RoHS Compliant

Version 1.0

MCP9700/9700A
MCP9701/9701A

Low-Power Linear Active Thermistor ICs

Features

- Tiny Analog Temperature Sensor
- Available Packages:
  - SC70-5, SOT-23-3, TO-92-3
- Wide Temperature Measurement Range:
  - -40°C to +125°C (Extended Temperature)
  - -40°C to +150°C (High Temperature) **(MCP9700,** SOT-23-3 and SC70-5 only**)**
- Accuracy:
  - ±2°C (max.), 0°C to +70°C **(MCP9700A/9701A)**
  - ±4°C (max.), 0°C to +70°C **(MCP9700/9701)**
- Optimized for Analog-to-Digital Converters (ADCs):
  - 10.0 mV/°C (typical) **(MCP9700/9700A)**
  - 19.5 mV/°C (typical) **(MCP9701/9701A)**
- Wide Operating Voltage Range:
  - V_{DD} = 2.3V to 5.5V **(MCP9700/9700A)**
  - V_{DD} = 3.1V to 5.5V **(MCP9701/9701A)**
- Low Operating Current: 6 µA (typical)
- Optimized to Drive Large Capacitive Loads

Typical Applications

- Hard Disk Drives and Other PC Peripherals
- Entertainment Systems
- Home Appliance
- Office Equipment
- Battery Packs and Portable Equipment
- General Purpose Temperature Monitoring

General Description

MCP9700/9700A and MCP9701/9701A sensors with Linear Active Thermistor Integrated Circuit (IC) comprise a family of analog temperature sensors that convert temperature to analog voltage.

The low-cost, low-power sensors feature an accuracy of ±2°C from 0°C to +70°C (MCP9700A/9701A) and ±4°C from 0°C to +70°C (MCP9700/9701) while consuming 6 µA (typical) of operating current.

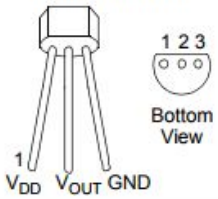
Unlike resistive sensors, e.g., thermistors, the Linear Active Thermistor IC does not require an additional signal-conditioning circuit. Therefore, the biasing circuit development overhead for thermistor solutions can be avoided by implementing a sensor from these low-cost devices. The Voltage Output pin (V_{OUT}) can be directly connected to the ADC input of a microcontroller. The MCP9700/9700A and MCP9701/9701A temperature coefficients are scaled to provide a 1°C/bit resolution for an 8-bit ADC with a reference voltage of 2.5V and 5V, respectively. The MCP9700/9700A output 0.1°C/bit for a 12-bit ADC with 4.096V reference.

The MCP9700/9700A and MCP9701/9701A provide a low-cost solution for applications that require measurement of a relative change of temperature. When measuring relative change in temperature from +25°C, an accuracy of ±1°C (typical) can be realized from 0°C to +70°C. This accuracy can also be achieved by applying system calibration at +25°C.

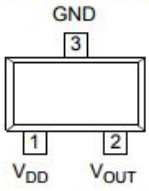
In addition, this family of devices is immune to the effects of parasitic capacitance and can drive large capacitive loads. This provides printed circuit board (PCB) layout design flexibility by enabling the device to be remotely located from the microcontroller. Adding some capacitance at the output also helps the output transient response by reducing overshoots or undershoots. However, capacitive load is not required for the stability of sensor output.

Package Types

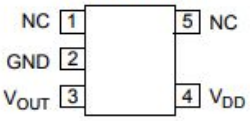
3-Pin TO-92
MCP9700/9700A
MCP9701/9701A



3-Pin SOT-23
MCP9700/9700A
MCP9701/9701A



5-Pin SC70
MCP9700/9700A
MCP9701/9701A



Micropower, 3-Axis, ±2 *g*/±4 *g*/±8 *g* Digital Output MEMS Accelerometer

Data Sheet **ADXL362**

FEATURES

- **Ultralow power**
  - **Power can be derived from coin cell battery**
  - **1.8 µA at 100 Hz ODR, 2.0 V supply**
  - **3.0 µA at 400 Hz ODR, 2.0 V supply**
  - **270 nA motion activated wake-up mode**
  - **10 nA standby current**
- **High resolution: 1 mg/LSB**
- **Built-in features for system-level power savings:**
  - **Adjustable threshold sleep/wake modes for motion activation**
  - **Autonomous interrupt processing, without need for microcontroller intervention, to allow the rest of the system to be turned off completely**
  - **Deep embedded FIFO minimizes host processor load**
  - **Awake state output enables implementation of standalone, motion activated switch**
- **Low noise down to 175 µg/√Hz**
- **Wide supply and I/O voltage ranges: 1.6 V to 3.5 V**
  - **Operates off 1.8 V to 3.3 V rails**
- **Acceleration sample synchronization via external trigger**
- **On-chip temperature sensor**
- **SPI digital interface**
- **Measurement ranges selectable via SPI command**
- **Small and thin 3 mm × 3.25 mm × 1.06 mm package**

APPLICATIONS

- **Hearing aids**
- **Home healthcare devices**
- **Motion enabled power save switches**
- **Wireless sensors**
- **Motion enabled metering devices**

GENERAL DESCRIPTION

The ADXL362 is an ultralow power, 3-axis MEMS accelerometer that consumes less than 2 µA at a 100 Hz output data rate and 270 nA when in motion triggered wake-up mode. Unlike accelerometers that use power duty cycling to achieve low power consumption, the ADXL362 does not alias input signals by undersampling; it samples the full bandwidth of the sensor at all data rates.

The ADXL362 always provides 12-bit output resolution; 8-bit formatted data is also provided for more efficient single-byte transfers when a lower resolution is sufficient. Measurement ranges of ±2 *g*, ±4 *g*, and ±8 *g* are available, with a resolution of 1 mg/LSB on the ±2 *g* range. For applications where a noise level lower than the normal 550 µg/√Hz of the ADXL362 is desired, either of two lower noise modes (down to 175 µg/√Hz typical) can be selected at minimal increase in supply current.

In addition to its ultralow power consumption, the ADXL362 has many features to enable true system level power reduction. It includes a deep multimode output FIFO, a built-in micropower temperature sensor, and several activity detection modes including adjustable threshold sleep and wake-up operation that can run as low as 270 nA at a 6 Hz (approximate) measurement rate. A pin output is provided to directly control an external switch when activity is detected, if desired. In addition, the ADXL362 has provisions for external control of sampling time and/or an external clock.

The ADXL362 operates on a wide 1.6 V to 3.5 V supply range, and can interface, if necessary, to a host operating on a separate, lower supply voltage. The ADXL362 is available in a 3 mm × 3.25 mm × 1.06 mm package.

FUNCTIONAL BLOCK DIAGRAM

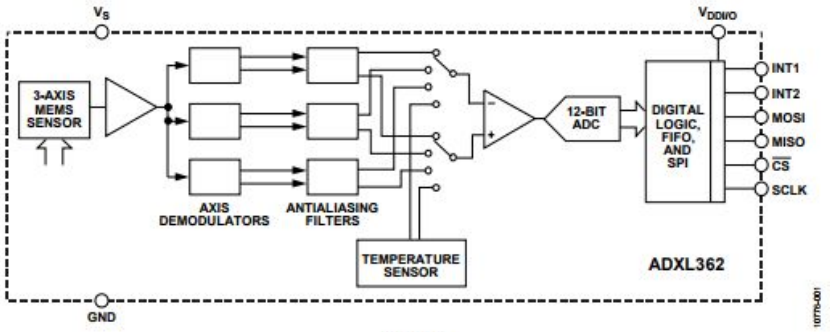


Figure 1.

Rev. E **Document Feedback**

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700
Technical Support www.analog.com