Final Report

Shreya Balaji
Rong Xiao

April 26, 2022

Introduction:

Our project is called the Wireless Sensor Node. The goal of this project is to wirelessly transmit data from one location to another via Bluetooth. There is a plethora of uses for such a system. Our system is built primarily to detect forest fires. By providing early warning signs and detecting the fire before it grows out of control, this system can help save many lives and prevent multiple tragedies. It can aid in protecting our natural resources and wild animals. This Wireless Sensor Node is apt to prevent all the wildfires occurring on the west coast.

When used to detect forest fires, the device will be placed in multiple locations in the forest. Hence, it must be able to run indefinitely without a power source. Using a thermistor, we read the temperature at a certain location. We will use a mock solar panel to send power to our circuit. A DC/DC converter will be used to increase the power from the Solar Panel to reach the level needed to charge the battery. This battery is what will power our Arduino which reads and transmits the temperature data. We used a PWM to regulate the DC/DC converter in order to save power. We went above and beyond the requirements by developing an Android app that monitored the temperature. This report will delve into the specific design process taken to develop this project.

High Level Design:

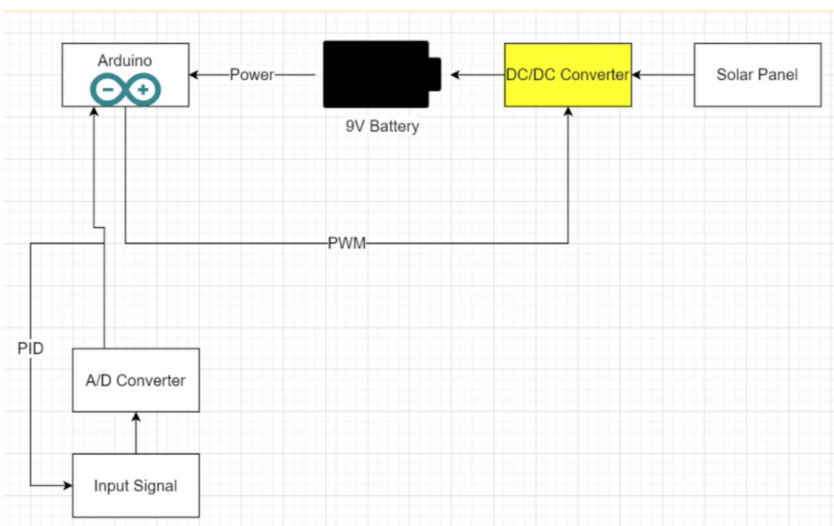

Figure 1: High Level Architecture

Above is our high-level design. It shows the connections between each subsystem and how the system works as a whole. Power will be taken from the solar panel and inputted into the DC/DC Converter. This converter will then amplify the power in order to charge the battery. The battery us then used to power the Arduino. The Arduino is what reads the temperature value and sends the data to the phone via Bluetooth. The Arduino is also responsible for monitoring the duty cycle of the converter, using PWM. The detailed design is provided below.

Detailed Design:

Hardware:

The DC/DC Converter is the main focus of our circuit. We began by deciding what type of converter we were going to use. After doing some research, we decided to build a boost converter. About 4V to 5V were being sent into our circuit via the mock solar panel. We need at least 8V to charge our battery. Hence, we chose a boost converter in order to amplify the voltage being sent in.

Once we picked the type of converter, we moved on to deciding which type of MOSFET and diode we wanted to use. After looking over the datasheets, we decided to use 1N4001 diode since it had the lowest voltage drop out of all the ones in our kit [1]. For our MOSFET, we used the 1RFZ44N model as it had the lowest threshold voltage [2].

After the diode and MOSFET was decided on, we moved to calculating the values of our capacitor and inductor. Below are the equations we used to solve for the inductor value.

$$I = V_g/(D'^2 R)$$

$$V_g = input\ voltage = 4V\ or\ 5V$$

$$D'^2 = 2.5\ or\ 2$$

$$R = internal\ resistance = 75m\Omega$$

Solving this equation, we get an inductor current of 10.6A to 13.3A. Next, we solved for the value of the inductor using the below equation.

$$\Delta iL = \left(\frac{V_g}{2L}\right) * DT_s$$

After performing our calculations, we found that the inductor value should be 500μH. Using the equation provided to us: $6.9\mu H/R^2$, we saw that we needed to wrap our wire 10 times around the inductor coil to achieve the wanted inductance.

Once our inductor was built, we moved to finding our capacitor value. We used the below equation to do so.

$$\Delta v = (V/2RC) * DT_s$$

Using a 10% ripple with 8.4V, we get 0.84V. However, we set $\Delta v$to be 0.1V since we want to be as efficient as possible. Using V = 5V and D = 40%, we see that the capacitor value is

33 μF . Since we do not have a 33 μF capacitor in our kit, we settled for the closest value, which is 47 μF .

Once our calculations were done, we moved to simulating our schematic. After some trials and testing, we added a few resistors to our circuit. These resistors mimic the load of the battery. Some components began to heat due to too much current being delivered to them. Hence, we added a few extra resistors to reduce the current.

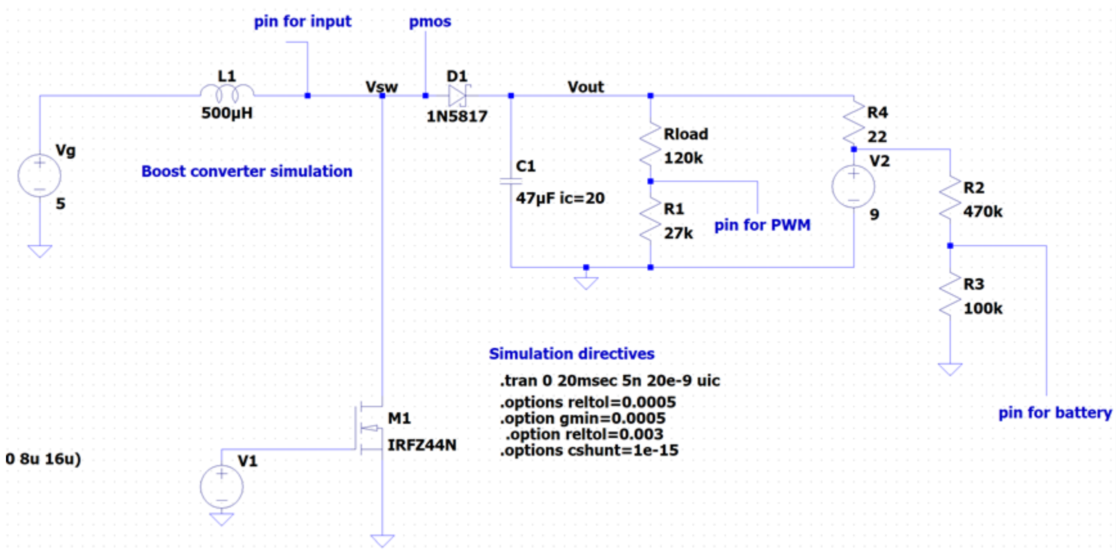

Figure 2: LTSpice Schematic

Software:

There are multiple software components to our project. The first component we will address is the PID portion. The flowchart is shown below, and the code is given in the appendix. As seen in the flowchart, our code reads the output voltage of our DC/DC Converter. Then, it compares that voltage to the desired voltage. Our code will then alter the duty cycle as necessary to regulate the voltage. A higher duty cycle means higher voltage and vice versa.

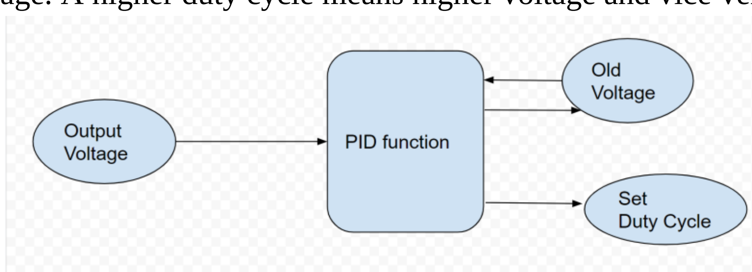

Figure 3: PID Software Flowchart

The next software component we will address is that for sleep mode. The flowchart for this code is given below and the actual code is provided in the appendix. We primarily used our watchdog timer to regulate sleep mode. The watchdog timer runs for 8 seconds at a time. If our processor is told it must go to sleep, it will first disable the ADC. Then it will power down and disable BOSD. Finally, the assembler as a whole will sleep. This process is repeated every time the processor is told it must go to sleep. We implemented sleep mode to help save power. If our circuit is on continuously, it will use a lot of power. By using sleep mode, we can make our circuit more efficient.

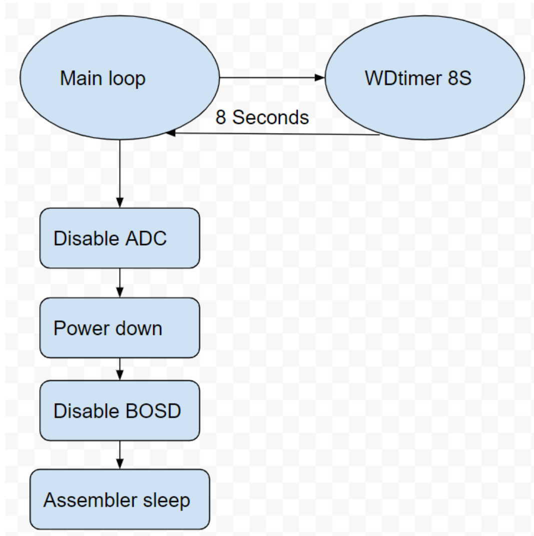

Figure 4: Sleep Mode Software Flowchart

The next component of our software is the Bluetooth module and temperature sensor. The flowchart for this code is given below and the actual code is provided in the appendix. First, we read the values from our thermistor. Since this value is not given in degrees Fahrenheit, we had to convert it using a formula provided in the thermistor data sheet [3]. This formula is shown below. We had to solve for T to get the temperature and substituted in the values for the constants provided in the datasheet.

$$B = ln(R/R_0)/(1/T - 1/T_0)$$

Once we had our value in degrees Fahrenheit, we used the serial.print function to output the data to the serial monitor. Then, we connected moved to MIT App inventor to work on the app portion of our project. In our app, we first added a Bluetooth enable connection. The user can click on the button and pair his/her phone to the Bluetooth module. If the connection is enabled, the Bluetooth will begin sending data to the phone. If the temperature goes above a certain value, a notifier is sent out, stating that there is a fire in the area. The user then has the option to either notify the authorities or cancel the notification. If they decide to notify the authorities, the case status will reflect as such.

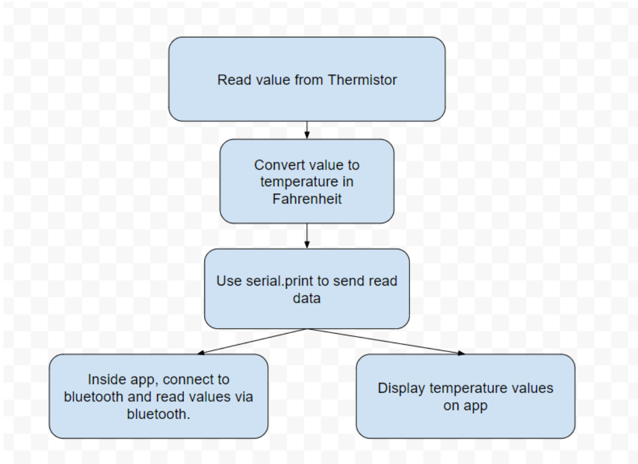

Figure 5: Thermistor Software Flowchart

Below is the flowchart for the entire software component of our project. We first start by initializing the Bluetooth connection and reading our temperature values. It specifies each of the feedback stages. The processor checks the input power of the circuit and decided whether to sleep, activate the converter, or not activate the converter.

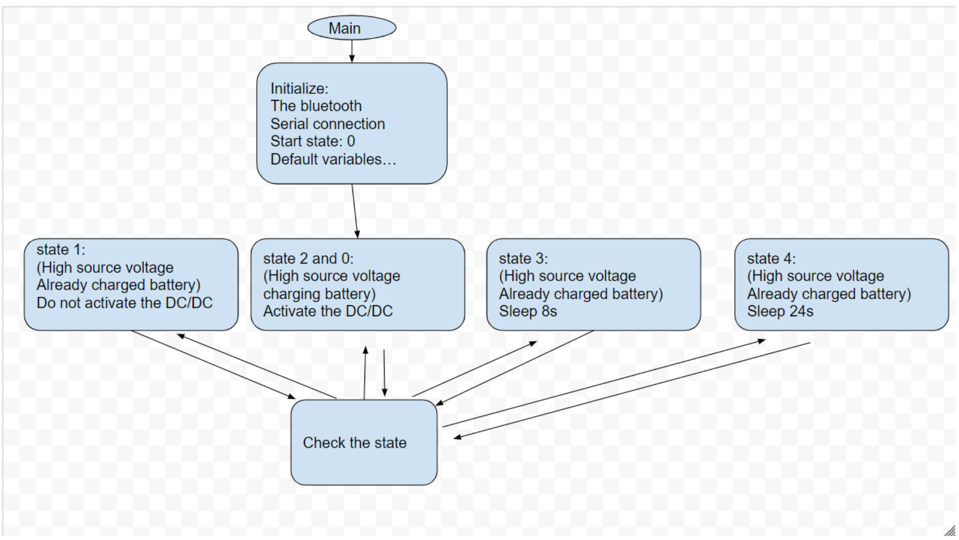

Figure 6: Overall Software Flowchart

Validation:

Hardware:

First, we worked on simulating our schematic. This helped us see if the values were roughly around where we wanted them to be. As shown in the below simulation screenshot, we can see that both the voltage and current start very high and then steady out. The current remains around 50mA, and the voltage is around 10V. This is exactly where we wanted our values to be in order to power our battery.

Figure 7: LTSpice Simulation

Once we confirmed that our simulation worked correctly, we moved forward to build the physical circuit. Below is our physical circuit. This includes the DC/DC Converter and the thermistor circuit. It took much testing and tweaking to build a working circuit.

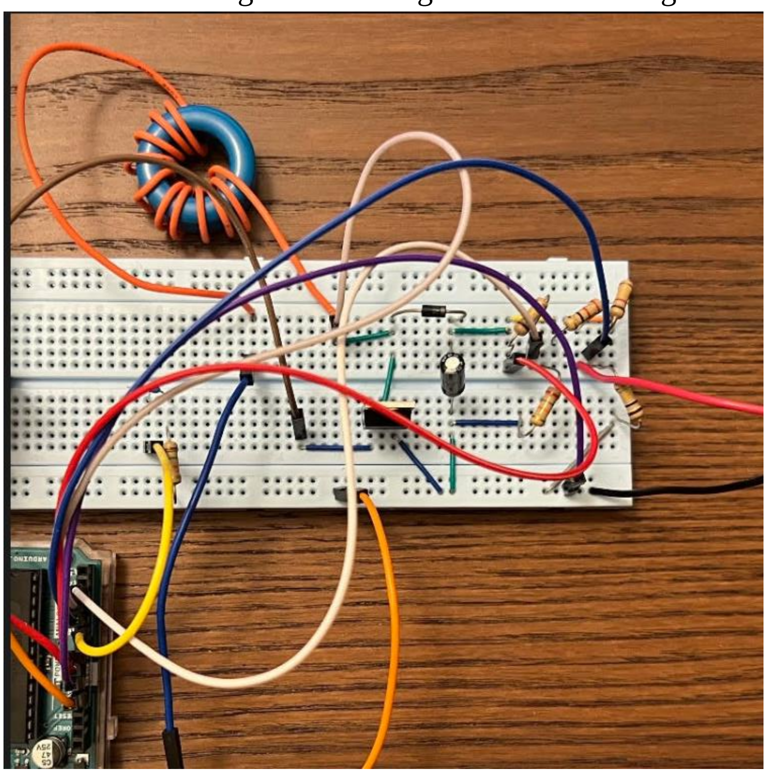

Figure 8: DC/DC Converter

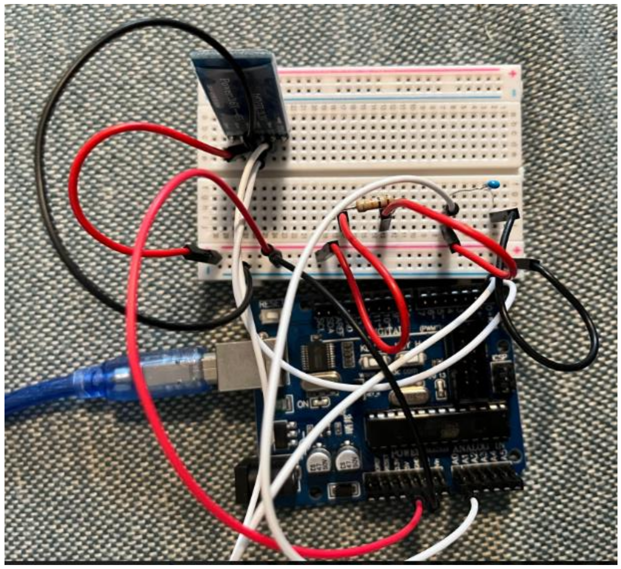

Figure 9: Thermistor and Bluetooth

Once we built our circuit, we worked on testing the voltage and current values that it outputted. As shown, our values read exactly what we had expected.

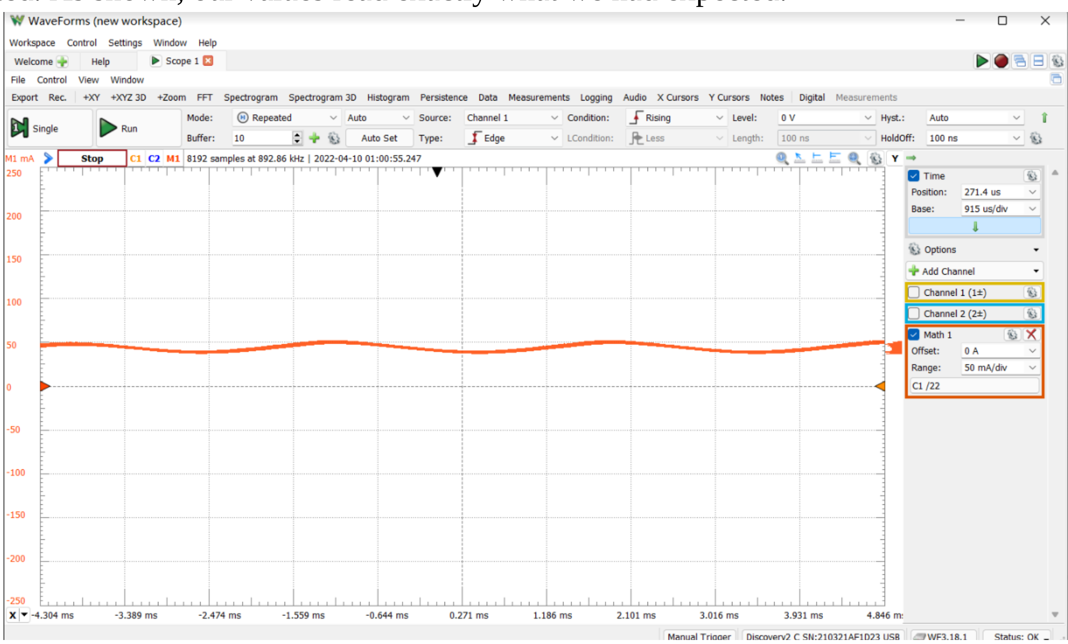

Figure 10: Circuit Output

We also tested the charging and discharging of our battery. Below are the plots of the data. These plots clearly show that the battery is charging and discharging as we had expected.

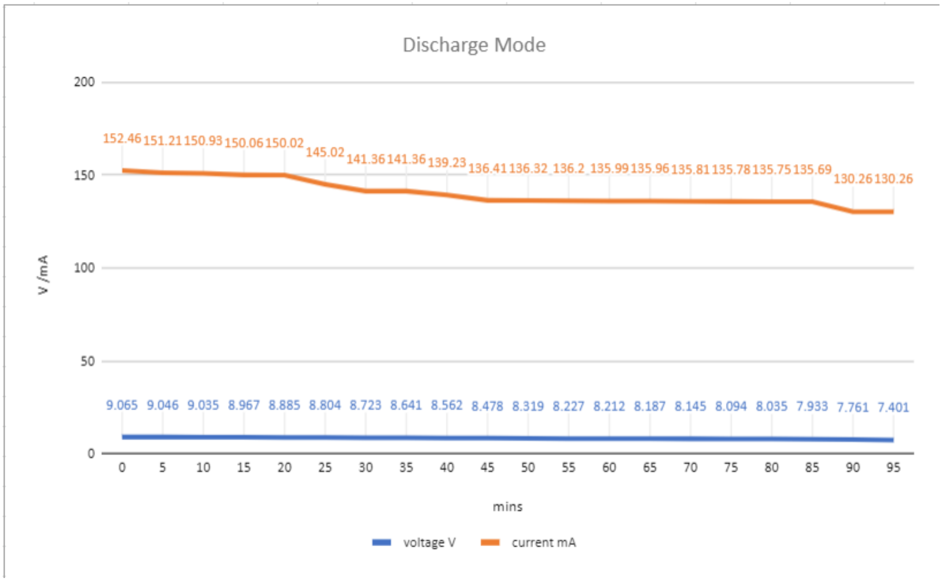

Figure 11: Discharging Waveform

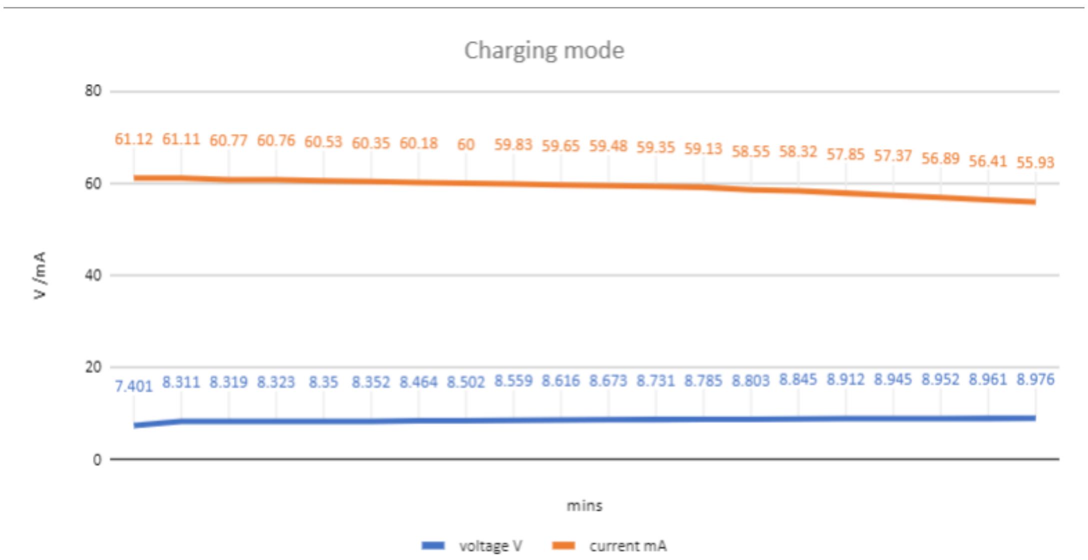

Figure 12: Charging Waveform

Through these tests, we were able to validate that our circuit was performing as expected.

Software:

Our PID code was tested when we measured the output of our code. Since our voltage and currents reading were reasonable, we knew that the duty cycle was being regulated to keep the voltage of the converter at bay. The next portion of our code that we tested was the power saving mode and sleeping mode. The image below shows how the processor goes to sleep every once in a while. This way, we can save power. When it wakes up, it also displays which stage it is currently in so that we know what power is being delivered to it. The underlines show the updating of the current stage of the processor.

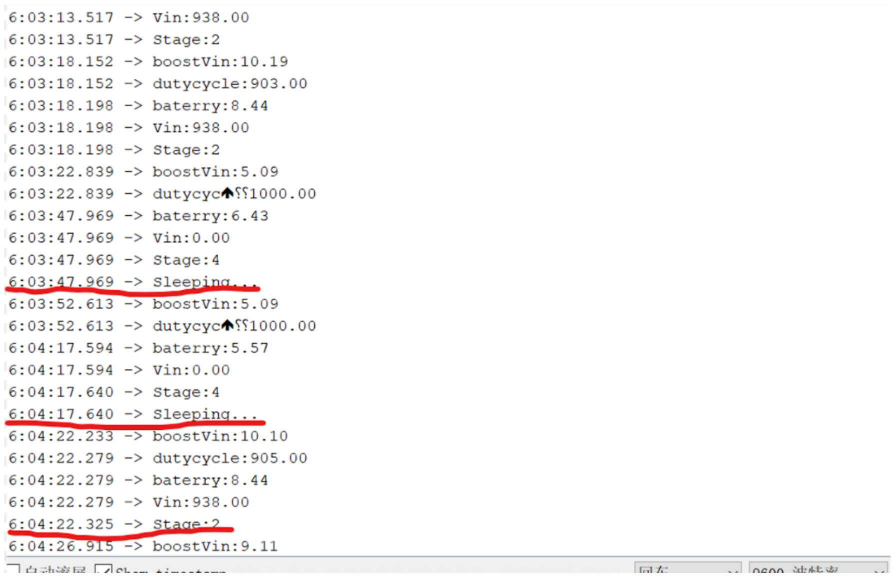

Figure 13: Serial Monitor Output

Next, we worked on testing our Bluetooth module and temperature sensor. First, we ran our code in the Arduino IDE and confirmed that our thermistor was outputting the correct values. We checked this by looking at the serial monitor. After we knew that the values were correct, we worked on testing our app. We did this by downloading the apk off of the MIT App Inventor platform onto an Android Phone. From there, we had to pair our Bluetooth module to the phone. We then went into the homepage of our app and were able to successfully read the temperature values from the thermistor, as shown below.

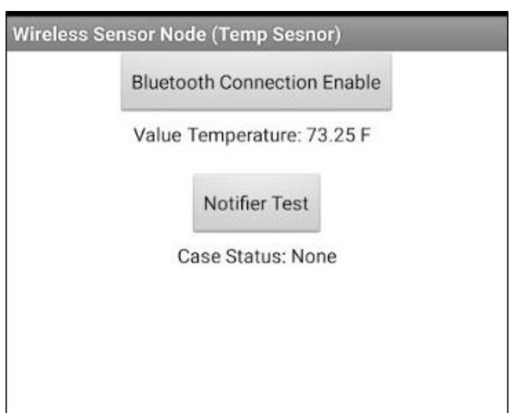

Figure 14: App Data 1

We used also blew a hairdryer onto the thermistor to increase the temperature. This was reflected on the app as expected. The result is shown below.

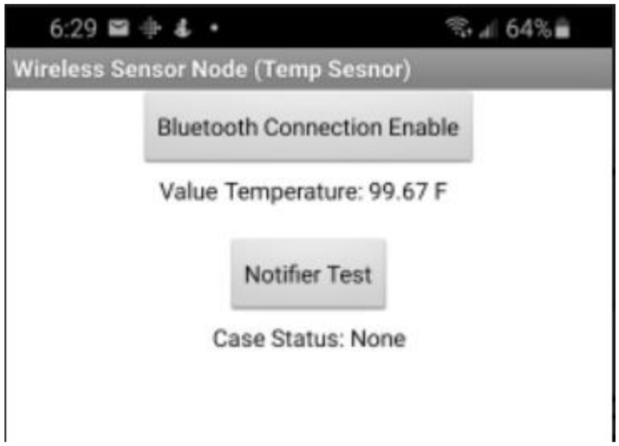

Figure 15: App Data 2

This proves that our app works as expected and is able to properly monitor the temperature values.

Conclusion:

In summation, we were able to produce a system that successfully reads and sends temperature data via Bluetooth. The phone app properly displays the temperature value and sends a notification if the temperature exceeds a certain value.

Throughout this process, we faced a plethora of challenges. The first challenge was that some of the components in our circuit were heating up too much. We ended up burning many of our components. In order to get through this, we added a few resistors to bring down the current. This took a lot of testing and going back and forth from the schematic and the physical circuit. Another challenge we faced was getting our Bluetooth to connect to the phone. We solved this problem by doing much research and testing various different codes and devices. However, we were able to successfully overcome these challenges and build a working project.

Throughout this project, we learned a multitude of lessons. We learned how to problem solve and learned to reach out if we needed help. We spent lots of time doing research and gaining a better understanding of our system. We learned to practice patience and break down our problems to find the root cause of our issues. We learned to work in a team and learned how to delegate tasks among members. If we were to do this project again, we would keep these imperative lessons in mind. The lessons we learned are extremely important and is something that we will carry with us throughout our lives. This project has been an immense learning experience for both of us.

Authorship:

Rong Xiao: I mainly contributed to the hardware and embedded system parts in this project. The main contributions were in the majority of the hardware design, including DC/DC Converter, voltage divider, and the assembly of the entire hardware project. My contribution to the software was the design of the entire hardware program structure, and the low power sleep program without any library.

Shreya Balaji: I contributed to this project by working on the software portion and the writing portion. I wrote the code for the PWM part, the Bluetooth module, and temperature conversion. I tested all of these parts and troubleshooted as necessary. I worked on building the app portion of the project and perfecting that. I also wrote all of the weekly reports, milestone reports, and final report, based on the images my partner provided of his work.

Appendix:

Code for Bluetooth and Thermistor:

```
//defining macros for constant values
#define RT0 10000   // 10000 ohms
#define B 3900      //Constant taken from data sheet
#define Vs 5    //Supply voltage
#define R 10000  //Resistance of thermistor is 10000 ohms

//defining variables
float RT, VR, lnRfrac, temp, T0, ThermistorOut;

void setup() {
  // put your setup code here, to run once:

  //setting baud rate
  Serial.begin(9600);
  //conversion of room temp from C to K
  T0 = 25 + 273.15;
}

void loop() {
  // put your main code here, to run repeatedly:


  //Reading value from thermistor
  ThermistorOut = analogRead(A0);
  //Converting value to voltage
  //1023 is max value of ADC
  ThermistorOut = (5.00 / 1023.00) * ThermistorOut;
  //Obtaining necessary values for VR and RT
  VR = Vs - ThermistorOut;
  RT = ThermistorOut / (VR / R);

  //Taking ln of r/r0
  lnRfrac = log(RT / RT0);

  //using overall equation to get obtain temperature in kelvin
  temp = (1 / ((lnRfrac / B) + (1 / T0)));
  //converting from K to C
  temp = temp - 273.15;
  //Converting from C to F
  temp = (temp*1.8) + 32;

  //Printing Values to Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.print(" F");
  Serial.println(' ');
  //adding a delay so that values are displayed at a readable speed
  delay(500);
}
```

Code for Sleep Mode:

28 lines (24 sloc) | 940 Bytes

```
//#include <avr/power.h>
//#include <avr/wdt.h>
//#include <avr/sleep.h>

void enterSleep(void){
  //set_sleep_mode(SLEEP_MODE_PWR_DOWN);//slect the mode
  MCUSR &= ~(1 << WDRF);              //flag
  WDTCSR |= (1<<WDCE) | (1<<WDE);     //enable configuration changes
  WDTCSR = (1<<WDP0) | (1<<WDP3);     //set 8s timer
  WDTCSR |= (1<<WDIE);                //enagble interrupt
  sleeping();
  //sleep_enable();                     //Enable the sleep
  //sleep_mode();                       //triger the sleep
  //time pass
  //sleep_disable();                    //wakeup
}


void sleeping(){
  ADCSRA &= ~(1<<7); //disable ADC
  SMCR |= (1<<2);    //power down
  SMCR |= 1;         // enable sleep
 //BOD diabled need befoe the asambler sleep
  MCUCR |=(3<<5);     //set the BODS and BODSE at the same time
  MCUCR = (MCUCR & ~(1<<5)) | (1<<6);//then sete them clear at the same time
  __asm__ __volatile__("sleep");//asambler sleep

}
```

Entire Combined Code:

```
// instiallizing the the loading library
//#include <avr/power.h>
//#include <avr/wdt.h>
//#include <avr/sleep.h>

//instializing the input pins
int HzOutPut =6;
double inputV = A3;
double BatteryV = A1;// not yet coding
double chargein = A2;// not yet coding

//this is the gloable values for the DC/DC converter
double boostVout = 0;
double dutysycle = 0;

//double RESISTER_VALUE_ONE = 30;
//double RESISTER_VALUE_TWO = 120;

//dutysycle instialize value
int DutyCycle = 50;

//DC intialize and charge target
double  setCharge = 9.2;
double  outDC= 0;
double  DCtemp = 0;

//timer
unsigned long myTime1;
unsigned long myTime2;

//Bettery voltage detect
double realBatteryV =0;


//voltage input
double  Vin = 0;

//instalize stage
int STAGE = 0;
int STAGE1 = 1;
int STAGE2 = 2;
int STAGE3 = 3;
int STAGE4 = 4;
```

```
//clock
double clocknumber = 0;

//int defult
int ZERO=0;

//WDT
volatile int f_wdt=0;

//temppreture defultes-----------------------------------------
#define RT0 10000   // 10000 ohms
#define B 3900      //Constant taken from data sheet
#define Vs 5    //Supply voltage
#define R 10000  //Resistance of thermistor is 10000 ohms

//defining variables
float RT, VR, lnRfrac, temp, T0, ThermistorOut;
//-----------------------------------------------------------------



void setup() {
  //Serial.begin(9600);
  initBluetooth();

  /*** Setup the WDT ***/

  /* Clear the reset flag. */
  MCUSR &= ~(1<<WDRF);

  /* In order to change WDE or the prescaler, we need to
   * set WDCE (This will allow updates for 4 clock cycles).
   */
  WDTCSR |= (1<<WDCE) | (1<<WDE);

  /* set new watchdog timeout prescaler value */
  WDTCSR = 1<<WDP0 | 1<<WDP3; /* 8.0 seconds */

  /* Enable the WD interrupt (note no reset). */
  WDTCSR |= _BV(WDIE);

  Serial.println("Initialisation complete.");
  delay(100); //Allow for serial print to complete.
TCCR0B = TCCR0B &11111000 | B00000001;//62kHz
pinMode(HzOutPut, outDC);
myTime2 = myTime1;

```

```
//---------------------------------------
  //conversion of room temp from C to K
  T0 = 25 + 273.15;
//-------------------------------------------
}
// main part
void loop() {
    f_wdt = 0;
  //updateSerial();
  //get the read from A3
  inputV = analogRead(A3);
  BatteryV = analogRead(A1);
  chargein = analogRead(A2);
  //


//-------------------------------------------
  // put your main code here, to run repeatedly:


  //Reading value from thermistor
  ThermistorOut = analogRead(A0);
  //Converting value to voltage
  //1023 is max value of ADC
  ThermistorOut = (5.00 / 1023.00) * ThermistorOut;
  //Obtaining necessary values for VR and RT
  VR = Vs - ThermistorOut;
  RT = ThermistorOut / (VR / R);

  //Taking ln of r/r0
  lnRfrac = log(RT / RT0);

  //using overall equation to get obtain temperature in kelvin
  temp = (1 / ((lnRfrac / B) + (1 / T0)));
  //converting from K to C
  temp = temp - 273.15;
  //Converting from C to F
  temp = (temp*1.8) + 32;

  //Printing Values to Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.print(" F");
  Serial.println(' ');
  //adding a delay so that values are displayed at a readable speed
  delay(500);
```

```
//   double  DCtemp = outDC;
//   outDC = DutyCycleCauculation(inputV, setCharge, DCtemp);
//   realBatteryV = BatteryV*0.0045*6.25*0.37;
  if (STAGE == 1){
      realBatteryV = BatteryV*0.0045*6.25*0.32;
      analogWrite(HzOutPut, ZERO);
      //digitalWrite(5, HIGH); // 5 to pmose
      clocknumber = 300000;
      //analogWrite(HzOutPut, ZERO);
  }
  if (STAGE == 2 || STAGE == 0){
      double  DCtemp = outDC;
      outDC = DutyCycleCauculation(inputV, setCharge, DCtemp);
      realBatteryV = BatteryV*0.0045*6.25*0.32;
      analogWrite(HzOutPut, outDC);
      //digitalWrite(5, LOW);
      clocknumber = 300000;
  }
  if (STAGE == 3){
    realBatteryV = BatteryV*0.0045*6.25*0.32;
//     ADCSRA &= ~(1<<ADEN);  // adc off
//     analogWrite(HzOutPut, ZERO);
//     delay(100);
//     clocknumber = 300000;
    //ADCSRA |= (1<<ADEN);
//     if (realBatteryV>0){
//       f_wdt = 0;
//     }
      if(f_wdt == 0)
  {
    enterSleep();
  }
  f_wdt = 0;
  STAGE = 2;
  }
  if (STAGE == 4){
    realBatteryV = BatteryV*0.0045*6.25*0.32;
    //ADCSRA &= ~(1<<ADEN);
        //delay(10000000);
    //analogWrite(HzOutPut, ZERO);
    clocknumber = 300000;
        Serial.println("Sleeping...");
        delay(100);
        //f_wdt=0;
              if(f_wdt == 0)
  {
    enterSleep();
```

```
    f_wdt = 0;
    STAGE = 2;
       // ADCSRA |= (1<<ADEN);
  }
  //analogWrite(HzOutPut, outDC);

    myTime1 = millis();
  //if(myTime1 - myTime2>= 300000*6.2){
  //if(myTime1 - myTime2>= 30000){
  if(myTime1 - myTime2>= clocknumber){
    myTime2 = myTime1;
    Serial.print("boostVin:");
    Serial.println(boostVout);
    Serial.print("dutycycle:");
    Serial.println(dutysycle);
    Serial.print("baterry:");
    Serial.println(realBatteryV);
    Serial.print("Vin:");
    Serial.println(chargein);

  if (chargein>1 && realBatteryV>9.2){
    STAGE = STAGE1;
  }
  if (chargein>1 && realBatteryV<9.2){
    STAGE = STAGE2;
  }
  if (chargein<1 && realBatteryV>7){
    STAGE = STAGE3;
  }
  if (chargein<1 && realBatteryV<7){
    STAGE = STAGE4;
  }
    Serial.print("Stage:");
    Serial.println(STAGE);
    //Serial.println(realBatteryV);
  }

}

double DutyCycleCauculation(double inputV, double setCharge, double preDC){
  double realVin = (inputV*0.0045)*5;

  boostVout = realVin;


   double out = preDC;
   if (realVin< setCharge && out<1000){
     out++;
   }
  else if(realVin > setCharge && out>0){
   out--;
  }
  dutysycle = out;

   return out;
 }

 //interrupt
 ISR(WDT_vect)
 {
   //realBatteryV = BatteryV*0.0045*6.25*0.32;
   if(f_wdt == 0)
   {
     f_wdt=1;
   }
 }
```

Resources:

[1]"HVCA 1N4001." https://www.alliedelec.com/product/hvca/1n4001/70015967/?gclid=EAIaIQobChMIk5ivm_b79QIVp9SzCh00pgIZEAAYASAAEgIFLPD_BwE&gclsrc=aw.ds. (accessed Feb. 15, 2022).

[2]B. Check, "Switched On: MOSFET Selection Guide," www.boulderes.com. https://www.boulderes.com/resource-library/switched-on-mosfet-selection-guide#:~:text=The%20lower%20the%20Qg%20of (accessed Feb. 15, 2022).

[3]"MOSFET Cross-Reference Search." https://alltransistors.com/mosfet/crsearch.php?&struct=MOSFET&polarity=N&pd=0.4&uds=60&id=0.2&rds=1.2&caps (accessed Feb. 15, 2022).

[4]"How to modify the PWM frequency on the arduino-part1(fast PWM and Timer 0) | eprojectszone," eprojectszone.com. https://www.eprojectszone.com/how-to-modify-the-pwm-frequency-on-the-arduino-part1/ (accessed Feb. 15, 2022).

[5]S. Ludin, A. Rahim, N. Mazalan, and S. Xia, "Design of DC-DC Boost Converter with LTSPICE Simulation Software," MIRJO, vol. 1, no. 2, pp. 20–27, Dec. 2016, Accessed: Feb. 15, 2022. [Online]. Available: http://maltesas.my/msys/explore/docs/2016/10_1482939560.pdf.

[6]"How do I simulate a dc-dc boost converter in LTspice?," Electrical Engineering Stack Exchange. https://electronics.stackexchange.com/questions/231064/how-do-i-simulate-a-dc-dc-boost-converter-in-ltspice (accessed Feb. 15, 2022).

[7]M. Mohammed, A. Ahmad, T. Abdulrahim, and Humod, "Efficiency Improvement of dc/dc Boost Converter by Parallel Switches Connection," International Journal of Applied Engineering Research, vol. 13, no. 9, pp. 7033–7036, 2018, Accessed: Mar. 29, 2022. [Online].

[8] "Getting Started with MIT App Inventor," appinventor.mit.edu. https://appinventor.mit.edu/explore/get-started.

[9] "Getting Started with HC-05 Bluetooth Module & Arduino," Arduino Project Hub, Apr. 19, 2019. https://create.arduino.cc/projecthub/electropeak/getting-started-with-hc-05-bluetooth-module-arduino-e0ca81.

[10] F. Hauke, "How to Receive Arduino Sensor-Data on Your Android-Smartphone," Instructables. https://www.instructables.com/How-to-Receive-Arduino-Sensor-Data-on-Your-Android/.

[11]H. Mathavna, "Get Sensor Data From Arduino To Smartphone Via Bluetooth," Electronics-Lab, Oct. 13, 2017. https://www.electronics-lab.com/get-sensor-data-arduino-smartphone-via-bluetooth/.

[12]“I need help with a runtime error,” MIT App Inventor Community, May 11, 2021. https://community.appinventor.mit.edu/t/i-need-help-with-a-runtime-error/32922/2 (accessed Apr. 12, 2022).
[13]J. Okerlund and F. Turbak, “Understanding App Inventor Runtime Errors.” Accessed: Apr. 12, 2022. [Online]. Available: https://cs.wellesley.edu/~tinkerblocks/summer13-okerlund.pdf
[14]“Live Development, Testing and Debugging,” ai2.appinventor.mit.edu. http://ai2.appinventor.mit.edu/reference/other/testing.html (accessed Apr. 12, 2022).
[15]“Troubleshooting for App Inventor 2,” appinventor.mit.edu. https://appinventor.mit.edu/explore/ai2/support/troubleshooting (accessed Apr. 12, 2022).
[16]“Unrecognized method. Irritants: (RunJavaScript),” MIT App Inventor Community, Mar. 26, 2021. https://community.appinventor.mit.edu/t/unrecognized-method-irritants-runjavascript/29685 (accessed Apr. 12, 2022).
[17]”Datasheet’ [Online] Available: https://www.arduino.cc/en/uploads/Tutorial/595datasheet.pdf
[18]“Shop 6F22 9V 280mah Rechargeable Batteries online,” EBLOfficial. https://www.eblofficial.com/products/9v-rechargeable-battery (accessed Feb. 15, 2022).
[19]“Highest PWM frequency output for the Uno/Nano,” Arduino Forum, Oct. 21, 2012. https://forum.arduino.cc/t/highest-pwm-frequency-output-for-the-uno-nano/125626 (accessed Feb. 22, 2022).
[20] “digikey.com,” 2 2022.. Available: https://www.digikey.com/en/products/detail/infineon-technologies/IRFZ44NPBF/811772.12 2 2022.
[21] “alltransistors.com,” alltransistors.com. Available: https://alltransistors.com/mosfet/crsearch.php?&struct=MOSFET&polarity=N&pd=0.4&uds=60&id=0.2&rds=1.2&caps=TO92. 11 2 2022.
[22] “boulderes.com,” boulderes.com. Available: https://www.boulderes.com/resource-library/switched-on-mosfet-selection-guide#:~:text=The%20lower%20the%20Qg%20of,a%20better%20choice%20over%20others.12 2 2022.
[23] “http://ltwiki.org/,” http://ltwiki.org/. Available: http://ltwiki.org/index.php?title=Standard.mos. 12 2 2022.
[24] “torexsemi.com,” torexsemi.com. Available: https://www.torexsemi.com/technical-support/tips/select-external-parts/. 2 2022.
[25] “elpro.org,” elpro.org. Available: https://www.elpro.org/gb/cdil-1n52b-series/113053-1n5226b.html. 2 2022.
[26] “alliedelec.com,” alliedelec.com. Available: https://www.alliedelec.com/product/hvca/1n4001/70015967/?gclid=EAIaIQobChMIk5ivm_b79QIVp9SzCh00pgIZEAAYASAAEgIFLPD_BwE&gclsrc=aw.ds. 2 2022.