University of California, Davis

Department of Land, Air and Water

Field Data-logger 1.0

Author:
Hengjiu Kang

July 20, 2015

Field Data-Loger 1.0

Hengjiu Kang*

Department of Electrical and Computer Engineering, University of California, Davis

July 20, 2015

*Electronic address: hjkang@ucdavis.edu

Figure 1: *Overall view of field Data-logger*

Contents

List of Figures

1 Overview

This Field Data-logger is a new hardware based on Atmega328 chip which fuses multiple type of sensors from different brands. It can collect data in programmable period of time, and thanks to low power design, this Field Data-logger can last for very long time when solar panel is installed.

2 Features

- 8 16-bit single ended or 4 16-bit differential ADC inputs
- 4 I^2C connection ports
- 4 Read only sensors inputs
- Programmable wake up RTC
- Solar panel
- On-board SD card
- XBee network (Will be available in the next version)

3 Hardware Design

3.1 Absolut values

Parameter	Sym	Min	Typ	Max	Unit	Conditions
External Battery	Vbat	3.2	3.8	4.2	V	
Digital Pin HIGH	Vhigh	3.3	3.3		V	
Digital Pin Low	Vlow	0	0	0.2	V	
Voltage Boost Output	5V	5.0	5.2	5.3	V	
ADC Sample rate	tADC		800		sps	
Boot time	tBoot		0.5		s	
5TE sensor cooling time			2		s	
ADC minimum input		-0.5			V	
ADC maximum input				5.5	V	
Default sleep period			60		min	
System sleep current			<100		uA	
System active current				45		

3.2 Electrical Systems

This field data-logger is designed in stack strategy to reach more compact size and more flexible connectivity. On the bottom is the main controller, middle shield is functionality shield, which has all the functional peripheral circuits on it, including external high precision ADC chip, power regulation circuit and level shifting circuit. Also, it has a 20-pin header pins to connect to the top shield, connection shield. Connection shield bears all the terminal connectors and simple protection circuits.

3.2.1 Controller board

Arduino Based controller Seeeduino Stalker v2.3 has been used as main controller. This controller is fully compatible with Arduino structure, and also it is able to connect to external Li-ion battery and a solar panel. According to our test, in UC Davis campus area, sun light is able to keep data-logger always on.

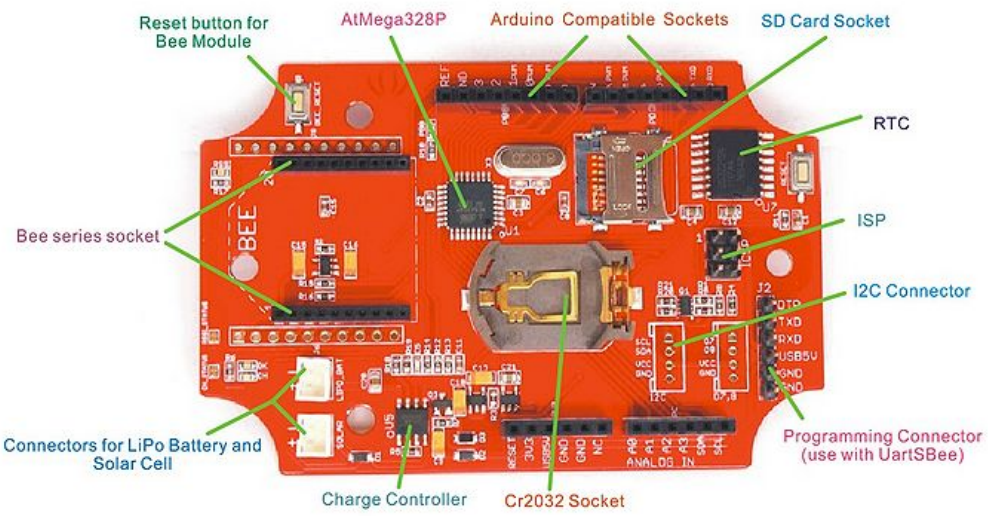

Figure 2: Seeeduino Stalker v2.3

3.2.2 Functionality board

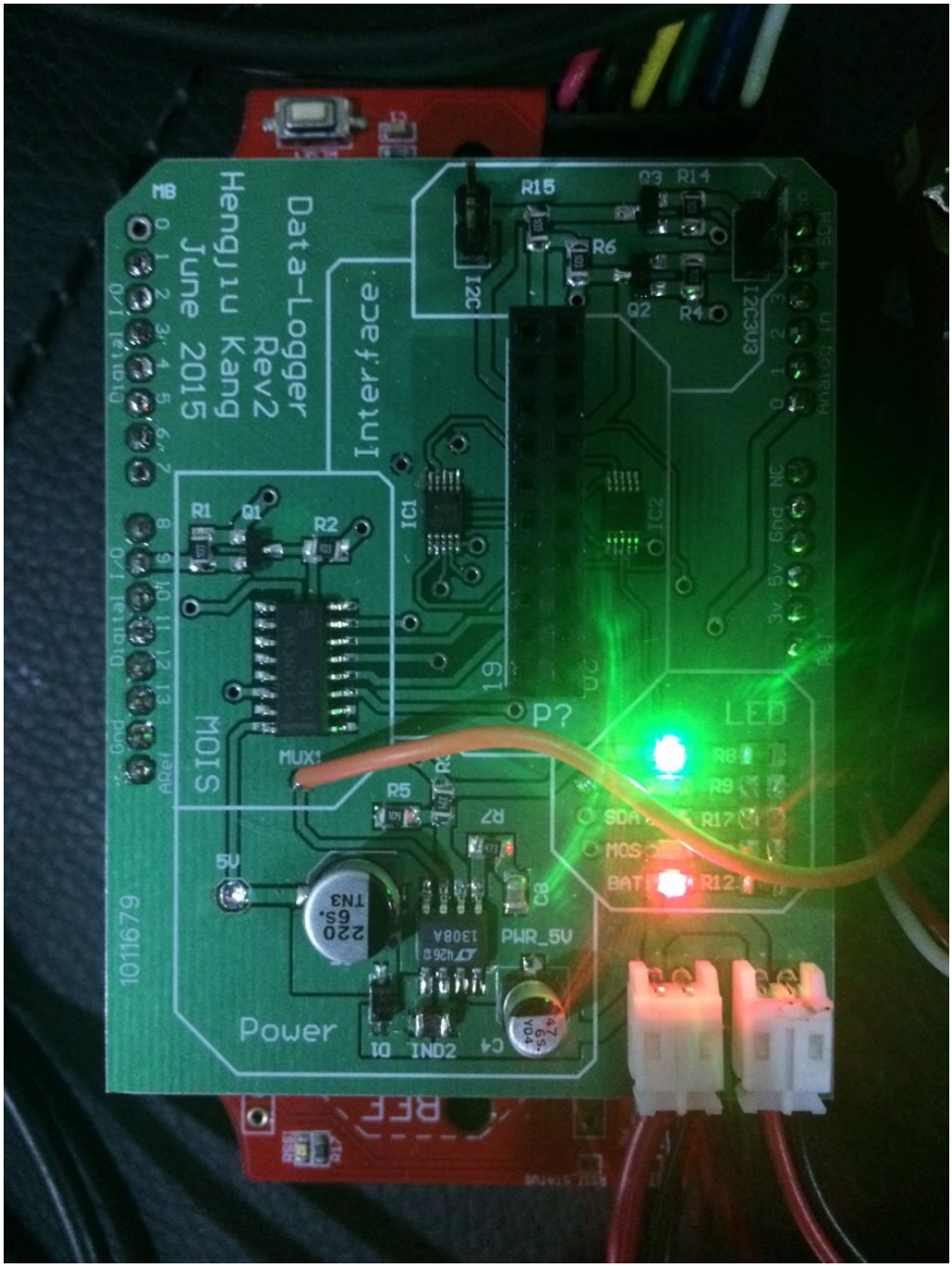

Figure 3: Functionality Board Top view

Functionality board is consisted of five functional blocks.

1. I^2C Level shifting blocks
   Because all the sensors are powered by 5V power source, so there is a 2-way

level shifting circuit here to convert I^2C connection. Refering to ***I2C Design*** for more information.

2. Interface blocks
   A 20-pin housing to connect to connection board.

3. MOIS blocks
   MOIS refers to 'Moisture sensor'. Default moisture sensor this data-logger uses is 5TE moisture sensor from decagon company, which uses 1-wire communication protocal, and there is a MUX chip let the board read at most 4 5TE sensors. Refering to ***Read-only Port*** for more information.

4. Power blocks
   Power block is circuit to convert battery voltage to 5V standard voltage powering all the devices. Refering to ***On-board power*** for more information.

5. LED blocks
   There are also five LEDs in this block for debugging purpose.

3.2.3 Connectivity board

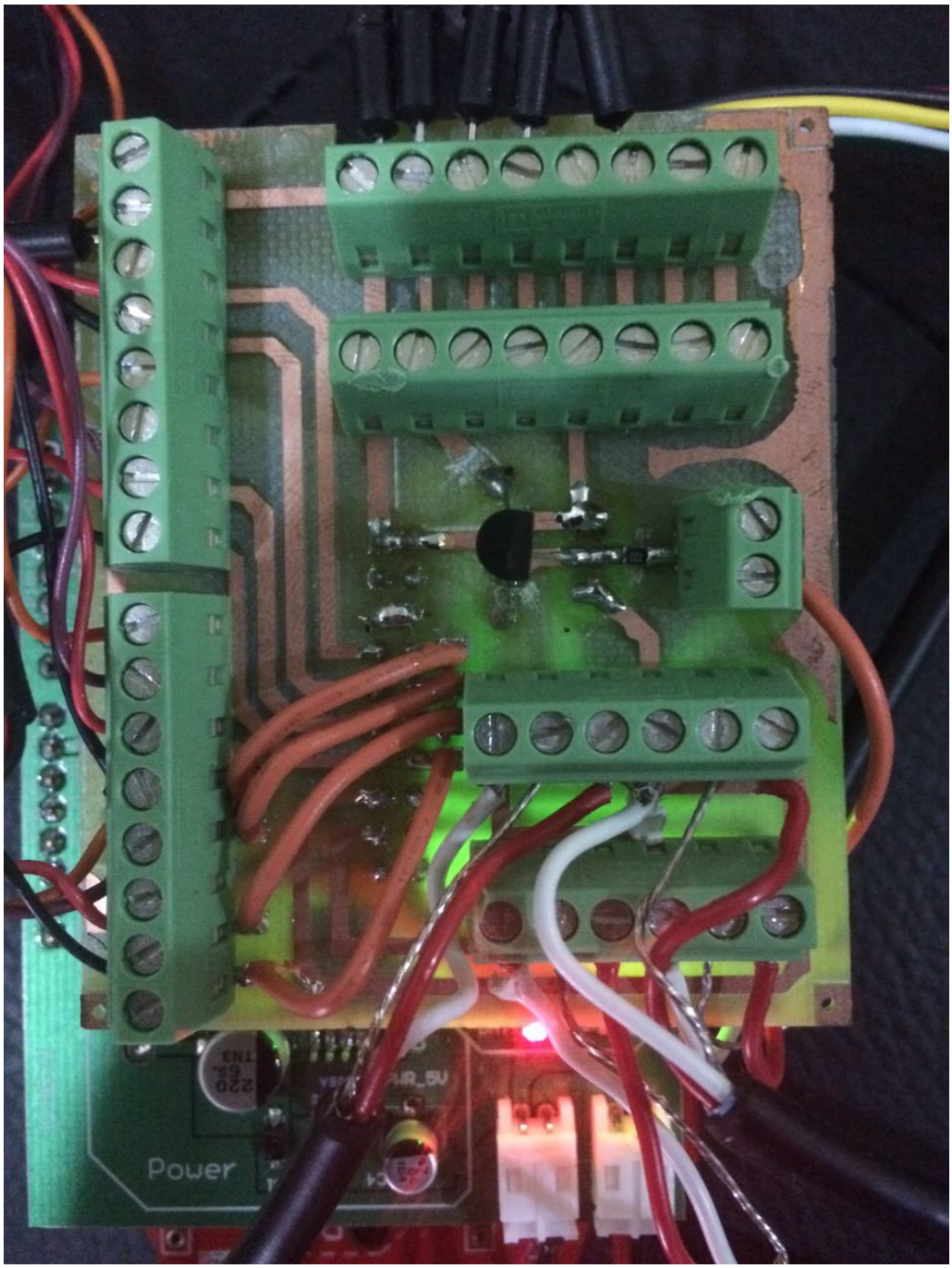

Figure 4: Connectivity Board Top view

Connectivity board has almost the same dimension of the functionality board and controller board, which is good to fit in the box. In order to increase the reliability of connection, 350 series terminal connector from Phoenix was chosen to make the job

done. All the connections are following the wire order :

- VCC
- GND
- connection 1 (SCL/ADC+)
- connection 2 (SDA/ADC-)
- etc..

3.3 Power

3.3.1 External power supply

Figure 5: Battery and solar panel

There are two possible power source: Li-ion battery and solar panel. In order to get enough power from battery, battery is directly connected to the functionality board, and a bridge wire is use to connect functionality board and controller board.

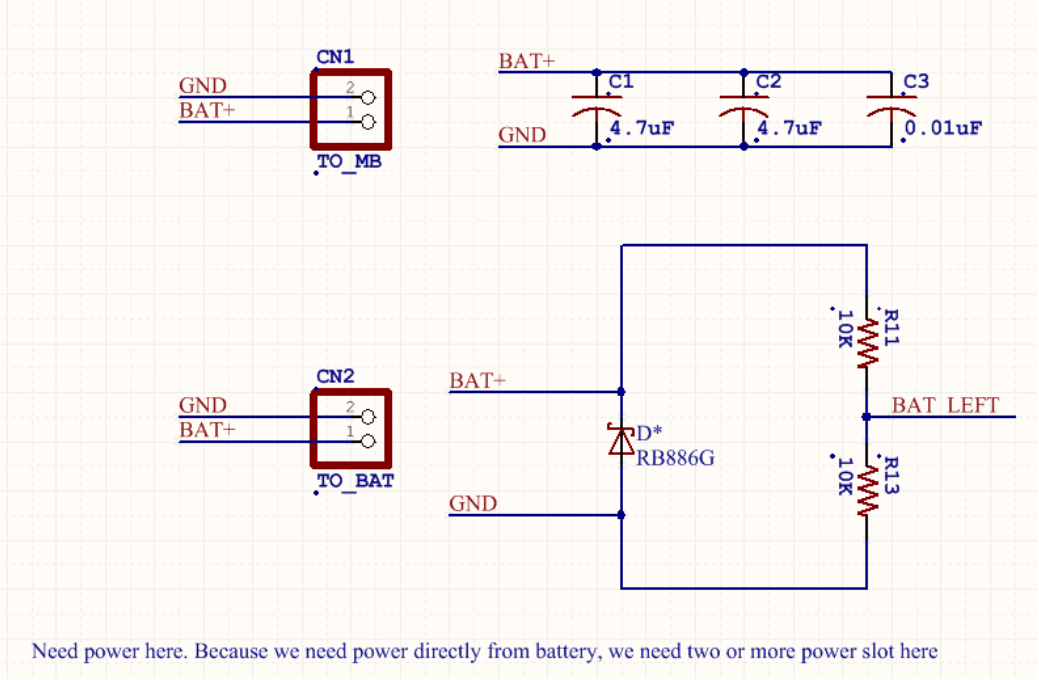

Figure 6: Power Bridge and coupling

3.3.2 On-board power

Seeeduino Stalker v2.3 has regulator to regulate external power supply to 3.3V standard, and all GPIO output is in 3.3V standard.

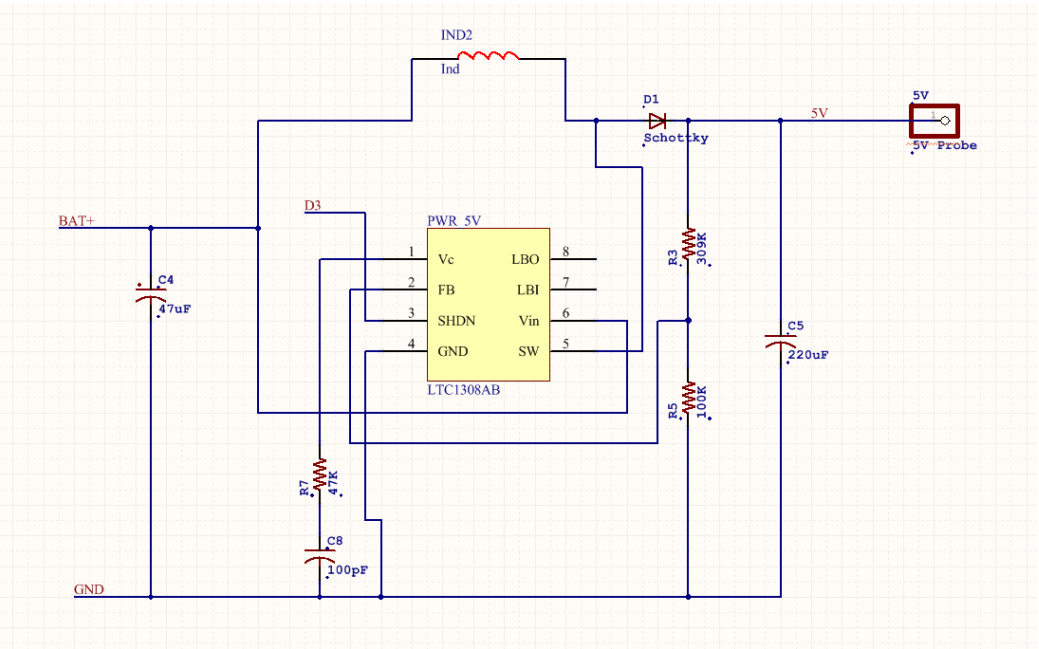

Figure 8: On board power boost circuit

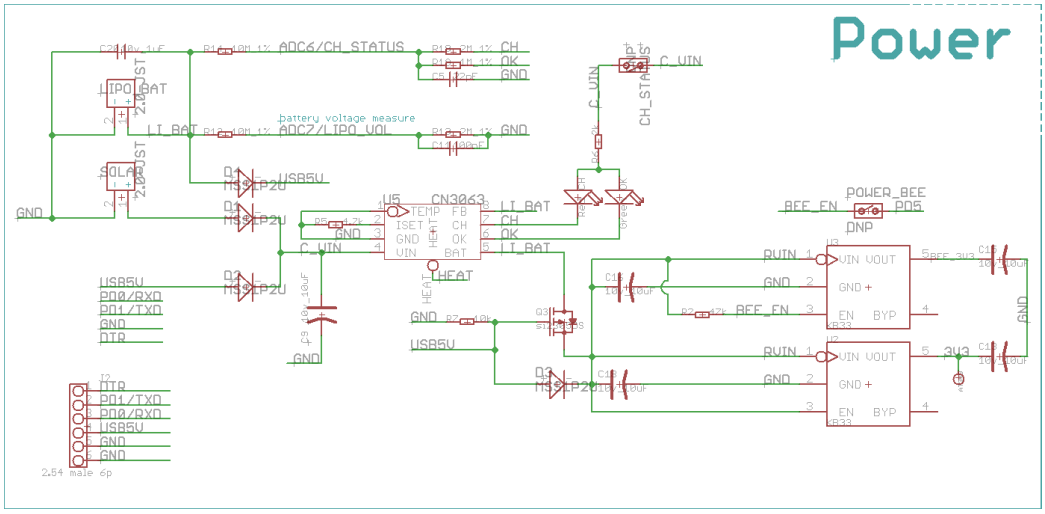

Figure 7: On board power regulator

In order to power on 5V standard peripheral devices, there is voltage boost regulator on the functionality board to boost battery voltage to 5V standard. This power boost chip's SHUTDOWN pin is controlled[1]by D5 pin on the controller board.

3.4 RTC

DS3231 chip is Extremely accurate $I^2C - intergrated$ RTC from Maximintegrated company. This data-logger has on-board DS3231 chip with backup button battery to keep timing, and wake up in period of time, which is excellent for extending battery life.
According to the offical application document from Seeeduino Wiki page, using RTC to wake up controller and put it into sleep mode can dramatically lower the power comsumption.

- The current consumption at sleep mode is 95.82 uA at 3.3V (i.e 316.206 uW power consumption). Please note, that the SD Card VCC is still powered in this demo.
- The current consumption at active mode peak is 22.43 mA @ 3.3V (i.e 74.019 mW power consumption)

[1]★**Known bug** Due to the design of voltage boost chip, when D5 pin is LOW, this boost circuit still gives out 3.2V output

3.5 I^2C Design

I^2C is important port in this data-logger, because devices inlcuding external ADC and precise temperature sensors are depending on I^2C to send data to the controller.

3.5.1 I^2C Port

I^2C Port on Seeeduino Stalker is signed to Analog pins area, a different from Arduino Uno.

Figure 9: I^2C port position on the Controller board

3.5.2 I^2C Level shifting

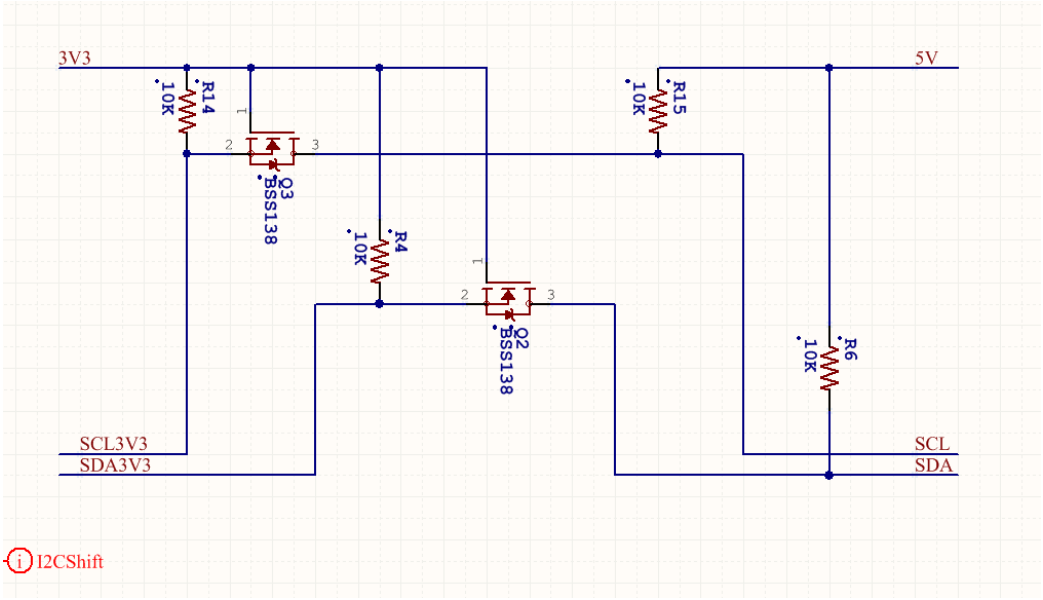

Figure 10: I^2C Level Shift circuit

Because all the external devices are working under 5V standard, so there is 2-way I^2C level shifting circuit[2]on the functionality board to convert 5V to 3.3V, verse versa.

[2]Thanks to the application document *Bi-directional level shifter for I2C-bus and other systems* from PHILIPS

3.6 Enhanced ADC

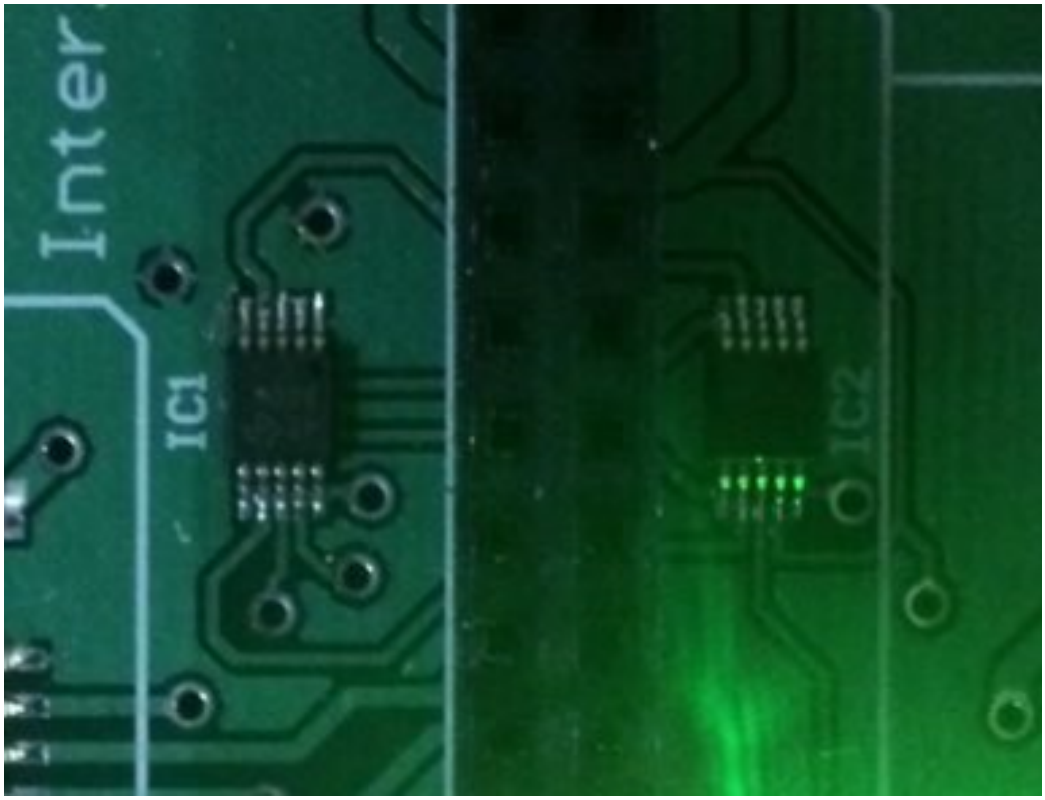

Figure 11: ADS1115 chip position on the functionality board

In order to eliminate noise, ADC chips are located as close as possible to the center connector.

Two ADS1115 16-bit high precision ADC chips from TI are used on the functionality board. These ADC circuits use I^2C protocal and give reliable rail-to-rail measuring ability to the data-logger. According to the design, 8 ADC ports are exposured to the outside world. User can either use them as 8 single-ended ADC inputs or 4 pair of differential ADC inputs.

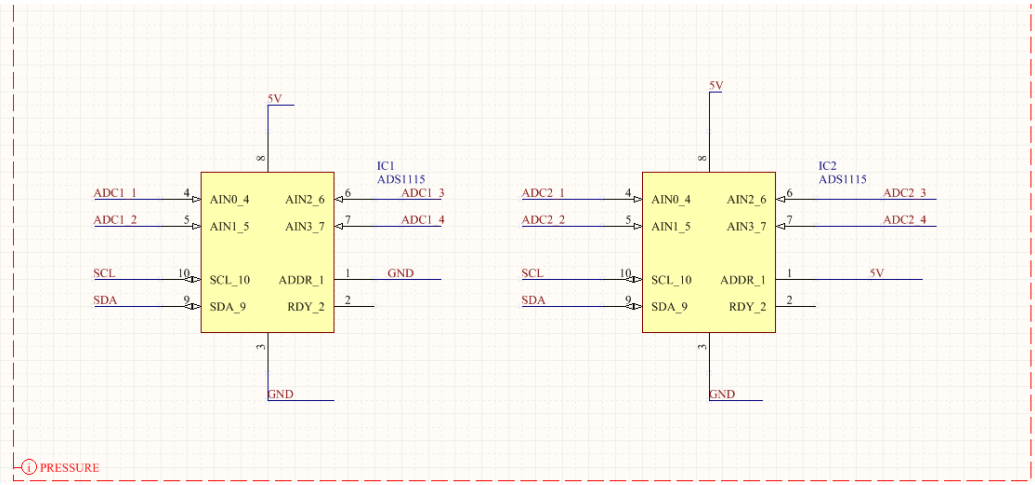

Figure 12: ADS1115 chip schematic

Two ADS1115 chips are configured with two I^2C address. No.1 chip is 0x48, and No.2 chip is 0x49. Refering to ***Absolute Value*** for more information.

3.7 Read-Only Port

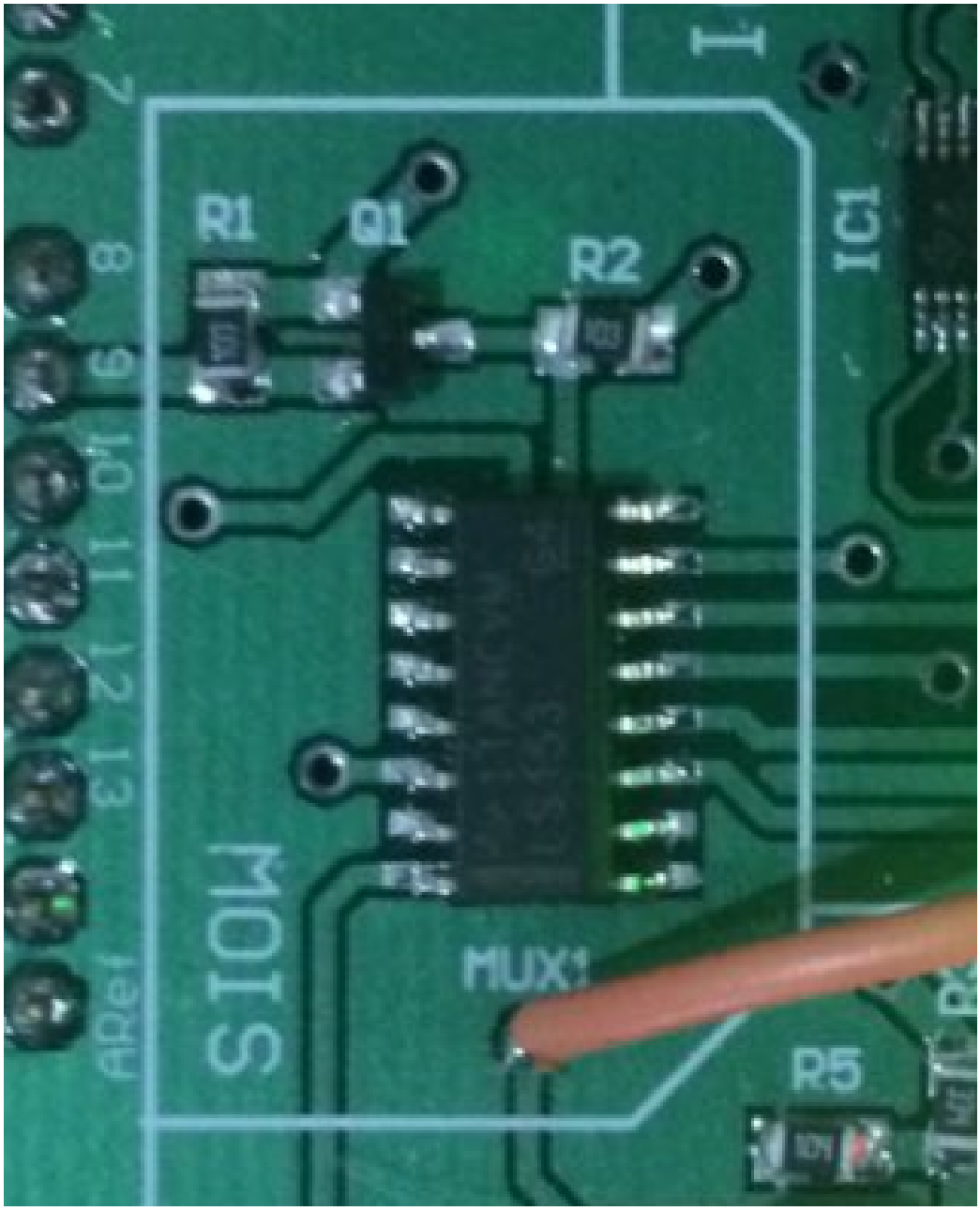

Figure 13: Read-Only port position on the board

Read Only port is implemented by a SoftSerial port on the controller and a 4-to-1 74x153 MUX chip with level shifter circuit, located on the side of functionality board.

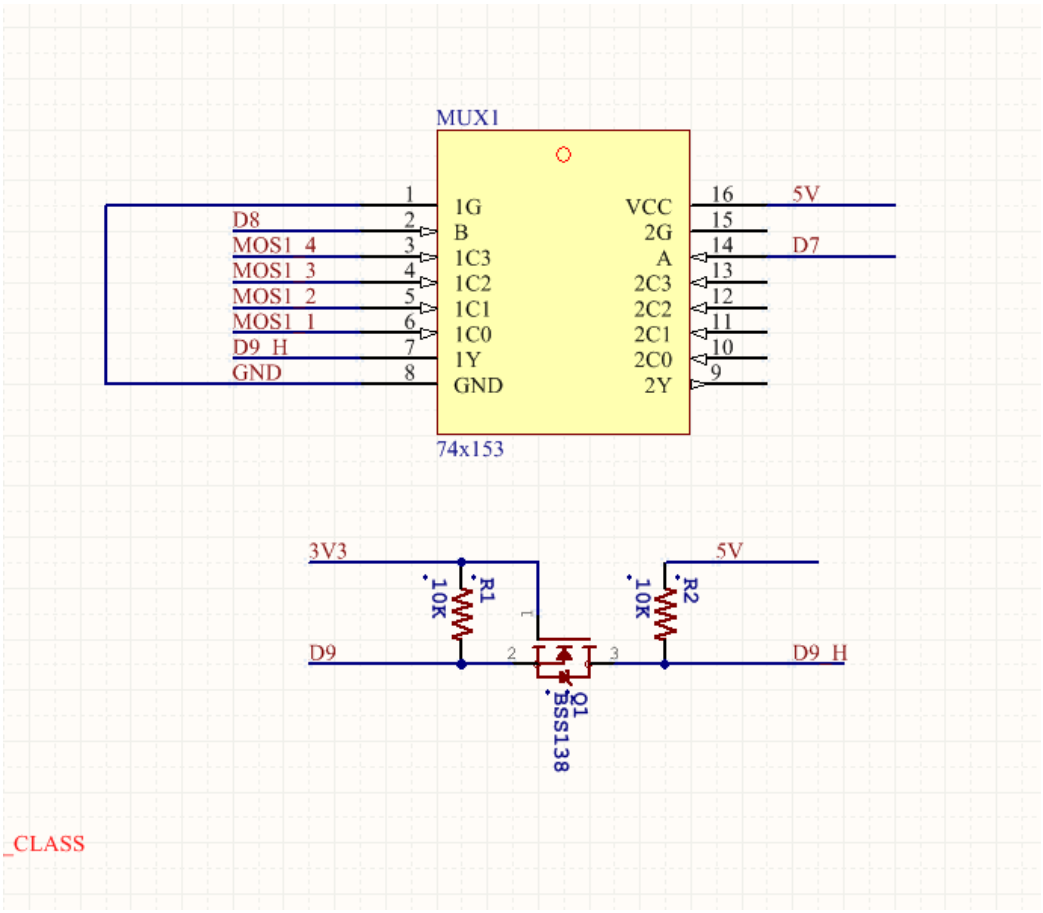

Figure 14: Read-Only port schematic

The moisture sensor, Decagon 5TE sensor uses 1-wire communication protocal which is compatible to Serial/UART port, so controller is able to read up to four 5TE sensors or other read-only sensors.
MUX has two select pins, which are D7 and D8. The output pin of group 1 is D9(5V), which will be converted to 3.3V standard, and this MUX is always enabled. The second group of 4-to-1 MUX on this 74x153 chip is not used.

3.8 Dimensions

Figure 15: Water-resistant box

All the parts, excluded connection wires can fit in a water-resistant box, which has dimension as 3.95in(100mm) X 2.68in(68mm) X 1.96in(50mm).

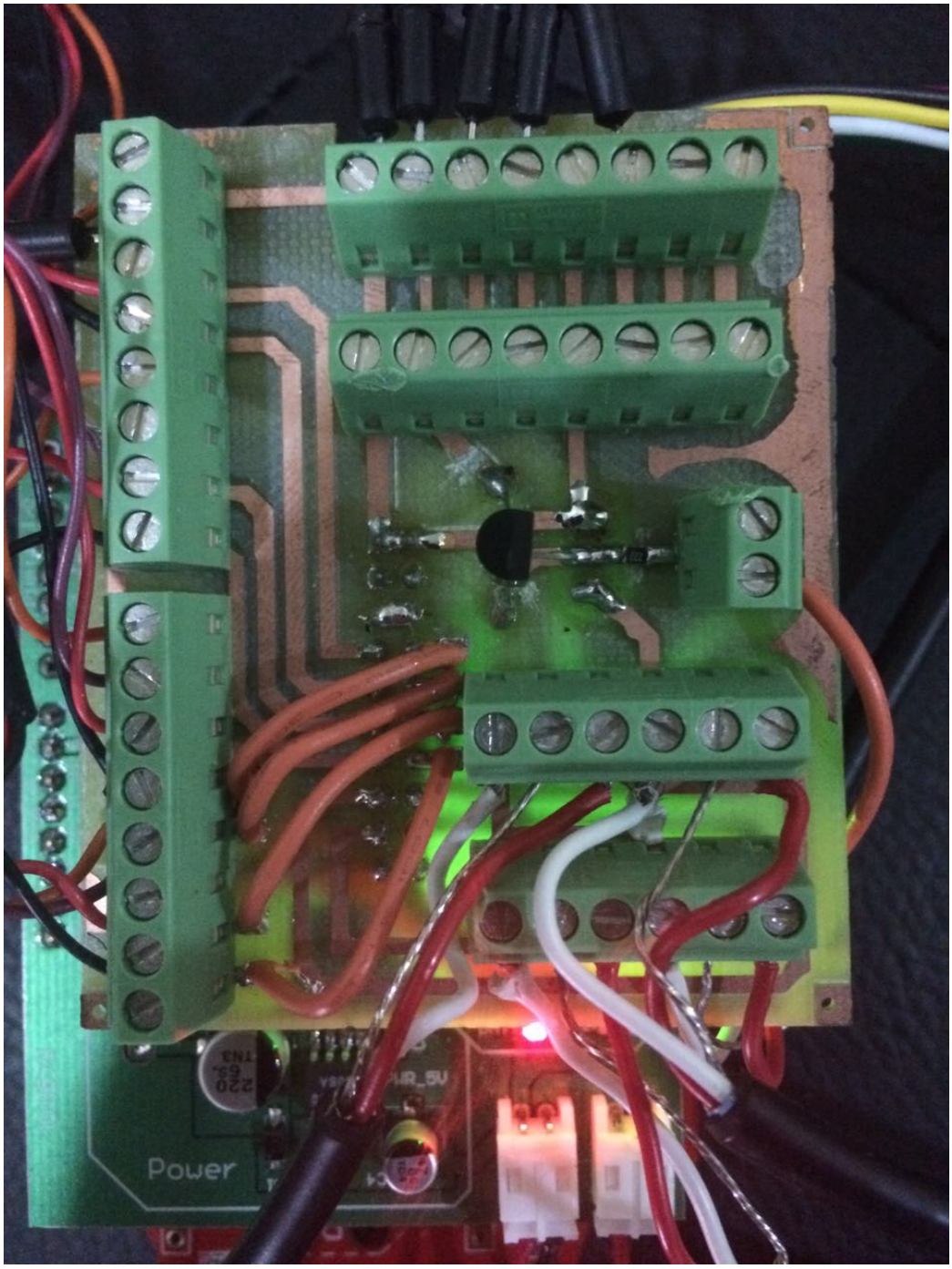

Figure 16: Connectivity Board Top view

All the parts, excluded connection wires can fit in a water-resistant box, which has dimension as 2.5in(63.5mm) X 2.1in(53.34mm) X 0.4in(10mm).

4 Software Design

4.1 Overall strucutre

All the components in this robot are written in C and optimized for power-saving consideration.
The following libraries are in use:

- ¡avr/sleep.h¿
- ¡avr/power.h¿
- ¡EEPROM.h¿
- ¡SdFat.h¿
- ¡stdio.h¿
- ¡Wire.h¿
- ¡SoftwareSerial.h¿
- ¡Adafruit_ADS1015.h¿
- ¡DS3231.h¿
- ¡Adafruit_MCP9808.h¿

And no customize library in use, so the firmware is better for general purpose usage.

5 Future Work

5.1 Wireless

Now the data-logger is in its release 1.0 version, and I am already started working on the 2.0 version, which is data-logger local network that has a coordinator to collect log files everyday them send them to the cloud.

Appendices

A PCB Design

A.1 Functionality board

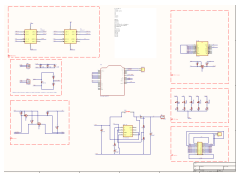

Figure 17: Functionality board PCB schematic

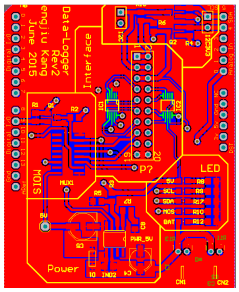

Figure 18: Functionality board PCB Layout

A.2 Connectivity board

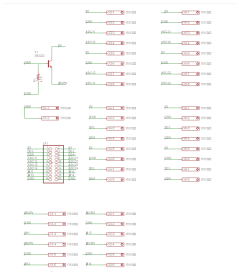

Figure 19: Connectivity board PCB schematic

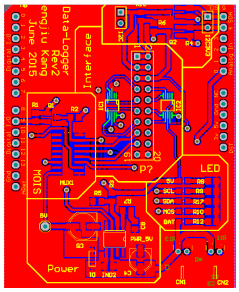

Figure 20: Connectivity board PCB Layout

B BOM

Due to limited space, please check individual BOM file.

C Source code

C.1 Applications

C.1.1 Main on Beaglebone Black

```
#include <avr/sleep.h>
#include <avr/power.h>
#include <EEPROM.h>
#include <SdFat.h>
#include <stdio.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <Adafruit_ADS1015.h>
#include "DS3231.h"
#include <Adafruit_MCP9808.h>




#define WAKE_UP   2
#define PWR_BOOST 3
#define SOFT_TX 4
#define XBEE_EN 5
#define MOI_PWR 6
#define MUX_A 7
#define MUX_B 8
#define MUX_Y 9
#define SD_CHIP_SELECT 10

#define MYID_ADD      (0x00)
#define MYID          (0x1A)

#define TEMP_MASK (B1111)
#define MOI_MASK (B1111)
#define PRESSURE_MASK (B1111)

SoftwareSerial mySerial(MUX_Y,SOFT_TX);
Adafruit_ADS1115 ADS1115A(0x48);
Adafruit_ADS1115 ADS1115B(0x49);
Adafruit_MCP9808 tempSensor = Adafruit_MCP9808();
```

```
// file system object
SdFat sd;
// text file for logging
SdFile myFile;
// Serial print stream
ArduinoOutStream cout(Serial);
DS3231 rtc;
DateTime nowTime;
DateTime lastTime;
char timeBuf[15];


//************************************void EEPROM_init
void EEPROM_init(){
  EEPROM.write(MYID_ADD,MYID);
}



void sleepNow()         // here we put the arduino to sleep
{

    set_sleep_mode(SLEEP_MODE_PWR_DOWN);   // sleep mode is set here

    sleep_enable();          // enables the sleep bit in the mcucr register

    attachInterrupt(0,INT0_ISR, LOW); // use interrupt 0 (pin 2) and run function
                                      // wakeUpNow when pin 2 gets LOW

    sleep_mode();            // here the device is actually put to sleep!!
                             // THE PROGRAM CONTINUES FROM HERE AFTER WAKING UP

    sleep_disable();         // first thing after waking from sleep:
                             // disable sleep...
    detachInterrupt(0);      // disables interrupt 0 on pin 2 so the
                             // wakeUpNow code will not be executed
                             // during normal running time.

}

void INT0_ISR()
{
  //Keep this as short as possible. Possibly avoid using function calls
    detachInterrupt(0);
}


//*************************MCP9808 Temperature********
float _temp9808_read(int choice) {
  // Read and print out the temperature, then convert to *F
   if (!tempSensor.begin(0x18+choice)) {
    return -255;
  }
  float c = tempSensor.readTempC();
  float f = c * 9.0 / 5.0 + 32;
  return c;
}

void noteDown_TEMP(){
  byte index=0;

   char fname[15];
  snprintf(fname, sizeof(fname), "%02d%02d%02d_T.log",
           nowTime.month(), nowTime.date(), nowTime.year()%100);

  if (!myFile.open(fname, O_RDWR | O_CREAT | O_AT_END)) {
    Serial.println("SD_temperature_error");
    return;
```

```
  }
  myFile.print(MYID,HEX);
  myFile.print(",");
  myFile.print(timeBuf);
  myFile.print(",");
  for (index=0;index<4;index++){
      myFile.print(_temp9808_read(index));
      myFile.print("␣");
  }
  myFile.println("");
  myFile.close();
}


//*************************record battery for reference********
int noteDown_BAT(){

  float voltage;
  int BatteryValue;
  BatteryValue = analogRead(A7);
  voltage = BatteryValue * (1.1 / 1024)* (10+2)/2;  //Voltage devider

  char fname[15];
  snprintf(fname, sizeof(fname), "%02d%02d%02d_B.log",
           nowTime.month(), nowTime.date(), nowTime.year()%100);

  if (!myFile.open(fname, O_RDWR | O_CREAT | O_AT_END)) {
    Serial.println("SD_BAT_error");
    return 1;
  }
  myFile.print(MYID,HEX);
  myFile.print(",");
  myFile.print(timeBuf);
  myFile.print(",");
  myFile.print(voltage);
  myFile.println("");
  myFile.close();
  return 0;
}

//*************************record pressure********
int noteDown_ADC_cell(Adafruit_ADS1115* ads1115){

  float multiplier = 0.0625F;
  int16_t results;
  int16_t results_1;
  delay(100);
      ads1115->begin();
      results=-ads1115->readADC_Differential_0_1();
      results_1=-ads1115->readADC_Differential_2_3();
  myFile.print(results*multiplier);
  myFile.print("␣");
  myFile.print(results_1*multiplier);
  return 0;
  }


int noteDown_ADC(){

  char fname[15];
  snprintf(fname, sizeof(fname), "%02d%02d%02d_P.log",
           nowTime.month(), nowTime.date(), nowTime.year()%100);


  if (!myFile.open(fname, O_RDWR | O_CREAT | O_AT_END)) {
    Serial.println("SD_pressure_error");
    return 1;
  }
```

```
  myFile.print(MYID,HEX);
  myFile.print(",");
  myFile.print(timeBuf);
  myFile.print(",");
  noteDown_ADC_cell(&ADS1115A);
  myFile.print("␣");
  noteDown_ADC_cell(&ADS1115B);
  myFile.println("");
  myFile.close();
 return 0;

}


//************************record moisture********
float TE_measure(byte* raw,int option){
  float mos=0;
  float cond=0;
  float temp=0;
  int i=0;
  while (raw[i]<'0' || raw[i]>'9'){
    i++;
  }
  while (raw[i]!='␣'){
   //     Serial.println(raw[i]-'0');
        mos=mos*10+raw[i]-'0';
        i++;
     }
  mos=mos/50.0;
  mos=4.3*pow(10,-6)*pow(mos,3)-5.5*pow(10,-4)*pow(mos,2)+2.92*pow(10,-2)*mos-5.3*pow
      (10,-2);
  //moisture

  i++;
  while (raw[i]!='␣'){
        cond=cond*10+raw[i]-'0';
        i++;
     }
  if(cond>700){
  cond=5*(cond-700)+700;
  }
  cond=cond/100;
  //conductivity

  i++;
  while (raw[i]!=0x0d){
        temp=temp*10+raw[i]-'0';
        i++;
     }
  if (temp>900){
  temp=5*(temp-900)+900;
  }
  temp=(temp-400)/10;
  switch (option){
    case 1:
      return mos;
    case 2:
      return cond;
    case 3:
      return temp;
    default:
      return 0;
  }
  return 0;
}

int noteDown_5TE_cell(int number){
  switch (number){
```

```
    case 1:
      digitalWrite(MUX_B,LOW);
      digitalWrite(MUX_A,LOW);
      break;
    case 2:
      digitalWrite(MUX_B,LOW);
      digitalWrite(MUX_A,HIGH);
      break;
    case 3:
      digitalWrite(MUX_B,HIGH);
      digitalWrite(MUX_A,LOW);
      break;
    case 4:
      digitalWrite(MUX_B,HIGH);
      digitalWrite(MUX_A,HIGH);
      break;
  }

  mySerial.flush();
  mySerial.listen();
  byte income;
  int i=0;
  byte raw[20];
  digitalWrite(PWR_BOOST,HIGH);
  delay(200);
  while (mySerial.available()){
    income=mySerial.read();
    raw[i]=income;
    i++;
    if (income==0x0A){
      break;
    }else if(i==19){
      break;
    }
  }
  digitalWrite(PWR_BOOST,LOW);


  myFile.print(TE_measure(raw,1));
  myFile.print("-");
  myFile.print(TE_measure(raw,2));
  myFile.print("-");
  myFile.print(TE_measure(raw,3));

  return 0;
}


int noteDown_5TE(){
  char fname[15];
  snprintf(fname, sizeof(fname), "%02d%02d%02d_M.log",
           nowTime.month(), nowTime.date(), nowTime.year()%100);


  if (!myFile.open(fname, O_RDWR | O_CREAT | O_AT_END)) {
    Serial.println("SD_moisture_error");
    return 1;
  }
  myFile.print(MYID,HEX);
  myFile.print(",");
  myFile.print(timeBuf);
  myFile.print(",");

  noteDown_5TE_cell(1);
  myFile.print("_");
  delay(1000);
  noteDown_5TE_cell(2);
  myFile.print("_");
```

```
  delay(1000);
  noteDown_5TE_cell(3);
  myFile.print("␣");
  delay(1000);
  noteDown_5TE_cell(4);
  myFile.print("␣");
  delay(1000);

  myFile.println("");
  myFile.close();
  return 0;
  }



//update time
void update_time(DateTime nowtime){
    snprintf(timeBuf, sizeof(timeBuf), "%02d%02d%04d␣%02d%02d",
            nowTime.month(), nowTime.date(), nowTime.year(),
            nowTime.hour(), nowTime.minute());
}


//send to coordinator
void Xbee_send_helper(char fname[],char cata){
  if (!myFile.open(fname, O_READ)) {
    Serial.println("SD␣send␣error");
    return;
  }
  char outchar[200];
  while (myFile.fgets(outchar,199,"\n")!=0){
    Serial.print(0xfe);
    Serial.print(MYID);
    Serial.print(cata);
    Serial.print(outchar);
    Serial.print(0xef);
  }
  myFile.close();


}
void Xbee_send(){
  digitalWrite(XBEE_EN,HIGH);
  delay(10000);  //waiting for log into network
  char fname[15];
  //pressure
  snprintf(fname, sizeof(fname), "%02d%02d%02d_P.log",
            lastTime.month(), lastTime.date(), lastTime.year()%100);
  if (sd.exists(fname)){
    Xbee_send_helper(fname,'P');
   }
  //moisture
  snprintf(fname, sizeof(fname), "%02d%02d%02d_M.log",
            lastTime.month(), lastTime.date(), lastTime.year()%100);
  if (sd.exists(fname)){
    Xbee_send_helper(fname,'M');
   }
  //temperature
  snprintf(fname, sizeof(fname), "%02d%02d%02d_T.log",
            lastTime.month(), lastTime.date(), lastTime.year()%100);
  if (sd.exists(fname)){
    Xbee_send_helper(fname,'T');
   }

   digitalWrite(XBEE_EN,LOW);
}
```

```
void setup(){
  Serial.begin(57600);
  //Serial.println("Hello World");
  EEPROM_init();
  //system
  analogReference(INTERNAL);
  //temperature
  //moisture
  pinMode(MUX_A,OUTPUT);
  digitalWrite(MUX_A,LOW);
  pinMode(MUX_B,OUTPUT);
  digitalWrite(MUX_B,LOW);
  pinMode(MUX_Y,INPUT);
  digitalWrite(MUX_Y,HIGH);

  pinMode(MOI_PWR,OUTPUT);
  digitalWrite(MOI_PWR,LOW);
  //pressure
  //power supply
  pinMode(PWR_BOOST,OUTPUT);
  digitalWrite(PWR_BOOST,LOW);
  //RTC
  pinMode(WAKE_UP, INPUT);
  digitalWrite(WAKE_UP,HIGH);
  rtc.begin();
  rtc.enableInterrupts(EveryHour);
  //SD card
  pinMode(SD_CHIP_SELECT,OUTPUT);
  if (!sd.begin(SD_CHIP_SELECT, SPI_HALF_SPEED)) sd.initErrorHalt();
  //xbee
  pinMode(XBEE_EN,OUTPUT);
  digitalWrite(XBEE_EN,LOW);

  delay(200);
  Wire.begin();
  mySerial.begin(1200);
  attachInterrupt(0,INT0_ISR, LOW);

}

void loop(){
  //Serial.println("Entered loop");
  delay(50);  //waiting for MCU stable
  rtc.clearINTStatus();
  nowTime = rtc.now();
  update_time(nowTime);


//   if (nowTime.hour()==12){     //newDay
 //    lastTime=nowTime;
//     Xbee_send();
 // }




//Serial.println("Record BAT");
  noteDown_BAT();

  //Serial.println("boost voltage");
  digitalWrite(PWR_BOOST,HIGH);
  delay(50);
  //Serial.println("Record Temperature");
  noteDown_TEMP();
  //Serial.println("Record ADC");
```

```
  noteDown_ADC();
  digitalWrite(PWR_BOOST,LOW);
  delay(500);
 // Serial.println("Record 5TE");
  noteDown_5TE();


  //delay(3000);
 sleepNow();
}
```