City, University of London

Department of Electrical and Electronic Engineering

LAB - MANUAL

EE3600 / EE3700 Design - III

Arduino Data Logger

T1 LAB - 6

1.0 Introduction

As engineers / developers we always rely upon the data collected to design or improve a system. Recording data and analysing them is a common practice in most of the industries, here we are building **Arduino Data Logger,** where we shall learn how we can log data at a specific interval of time. We will use an Arduino board to read some data (here temperature, humidity, date and time) and save them on a SD card and the computer simultaneously.

2.0 Required Components

- Arduino UNO
- DHT11 Sensor
- DS3231 RTC Module
- SD Card Module
- SD Card
- Connecting Wires
- Breadboard
- Computer

3.0 Component Description

3.1 DS 3231 RTC Module

When we come across an idea where keeping time is a prime concern, DS 3231 precision RTC module is a saviour. It is perfect for projects containing data-logging, clock-building, time stamping, timers, and alarms.

At the heart of the module is a low-cost, extremely accurate RTC chip. It manages all timekeeping functions and features a simple two-wire I2C interface which can be easily interfaced with any microcontroller platform of our choice.

The chip maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year (valid up to 2100).

The clock operates in either the 24-hour or 12-hour format with an AM / PM indicator. It also provides two programmable time-of-day alarms.

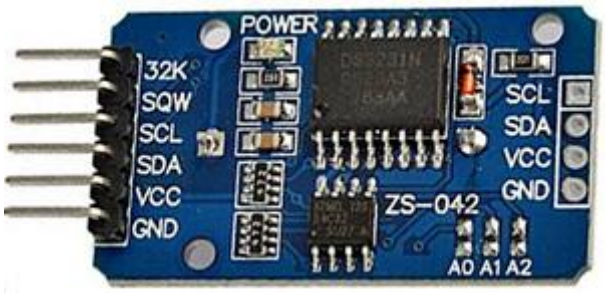

Figure1: DS 3231 module front view

The other feature of this board comes with SQW pin, which outputs a nice square wave at either 1Hz, 4kHz, 8kHz or 32kHz and can be handled programmatically. This can further be used as an interrupt due to alarm condition in many time-based applications.

The DS 3231 incorporates a battery input and maintains accurate timekeeping when main power to the device is interrupted.

The built-in power-sense circuit continuously monitors the status of VCC to detect power failures and automatically switches to the backup supply. So, you do not need to worry about power outages, your MCU can keep track of time.

Figure2: DS 3231 module rear view

The bottom side of the board holds a battery holder for 20mm 3V lithium coin cells. A CR2032 battery can fit well.

DS 3231 RTC module also comes with a 32 bytes 24C32 EEPROM chip from Atmel having unlimited read-write cycles. It can be used to save settings or really anything.

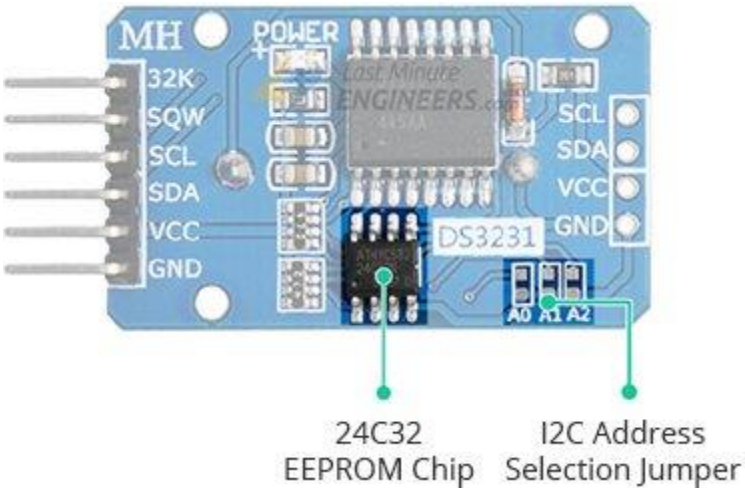

Figure3: DS 3231 chip and jumpers

The 24C32 EEPROM uses I2C interface for communication and shares the same I2C bus as DS3231.The I2C address of the EEPROM can be changed easily with the three A0, A1 and A2 solder jumpers at the back. Each one of these is used to hardcode in the address. If a jumper is shorted with solder, that sets the address.

Figure4: Jumpers

As per the 24C32's datasheet, these 3 bits are placed at the end of the 7-bit I2C address, just before the Read/Write bit.

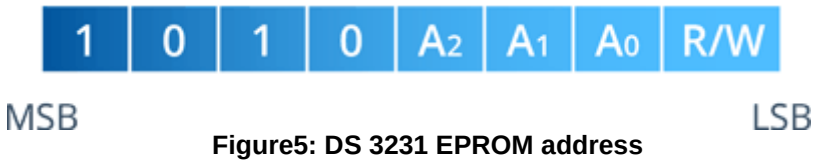

Figure5: DS 3231 EPROM address

As there are 3 address inputs, which can take 2 states, either HIGH / LOW, we can therefore create 8 (2^3) different combinations(addresses).

The DS 3231 RTC module has total 6 pins that interface it to the outside world. The connections are as follows:

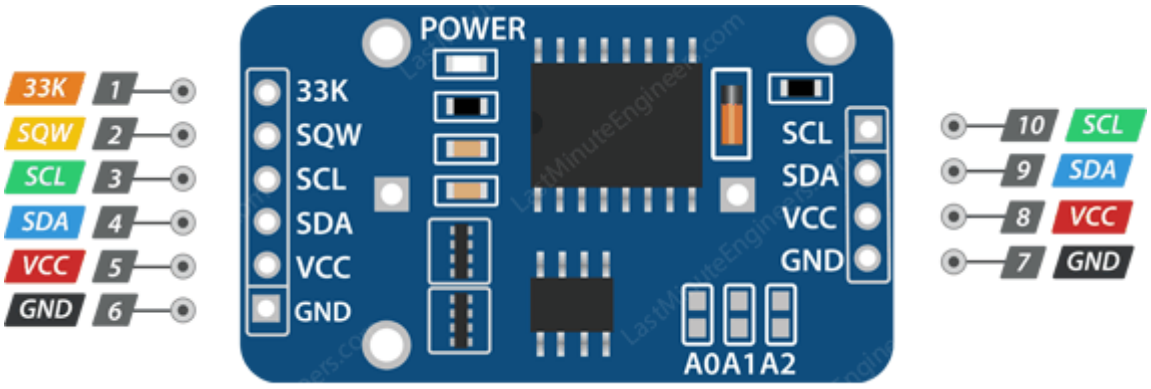

Figure6: DS 3231 Pin layout

32K pin outputs the stable (temperature compensated) and accurate reference clock.
SQW pin outputs a nice square wave at either 1Hz, 4kHz, 8kHz or 32kHz and can be handled programmatically. This can further be used as an interrupt due to alarm condition in many time-based applications.
SCL is a serial clock pin for I2C interface.
SDA is a serial data pin for I2C interface.
VCC pin supplies power for the module. It can be anywhere between 3.3V to 5.5V.
GND is a ground pin.

3.2 SD Card Module

A Micro SD Card is a flash based, removable memory device. It is non-volatile memory and is often used in mobile phones and other consumer electronic devices.

An SD Card Module or a Micro SD Card Adapter is a simple board which facilitates connection between a micro SD card and a microcontroller platform like Arduino.

Micro SD Card operates at 3.3V, a typical Micro SD Card Adapter or an SD Card Module basically consists of two important components. They are the 3.3V Voltage regulator IC and a 5V to 3.3V level converter IC for the communication pins.

Talking about pins, as I have mentioned that a Micro SD card supports only SPI communication, the SD Card Module has pins for SPI Communication. So, the pins on an SD card module are as follows. The following is the image of a typical SD Card Module.

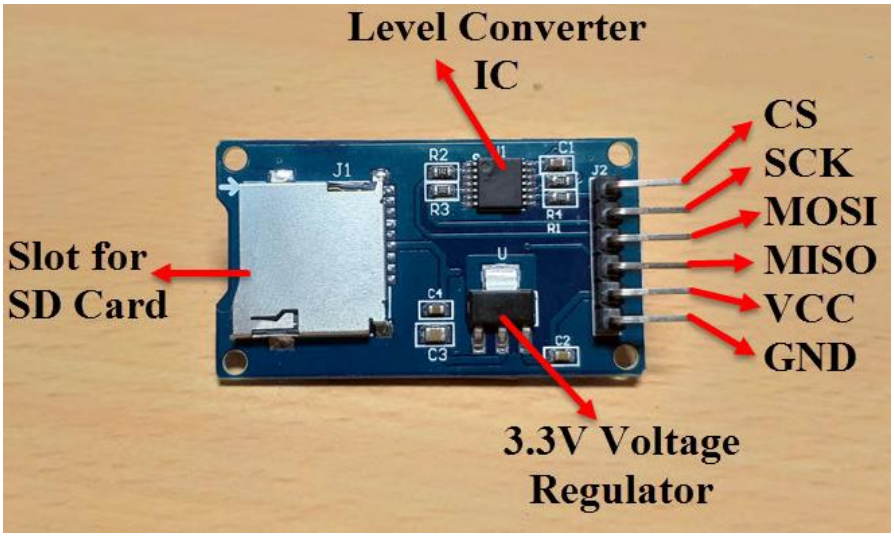

Figure7: SD card module front view

- VCC – 5V
- GND – GND
- MOSI – Master OUT Slave IN (Input)
- MISO – Master IN Slave OUT (Output)
- SCK – SPI Clock (Input)
- CS – Chip Select (Input)

4.0 Working of the Project

In this design project, we are going to learn how to use an SD card, RTC and a DHT11 temperature and humidity sensor to read the ambient temperature and

humidity saves these values in a SD card Microsoft Excel.

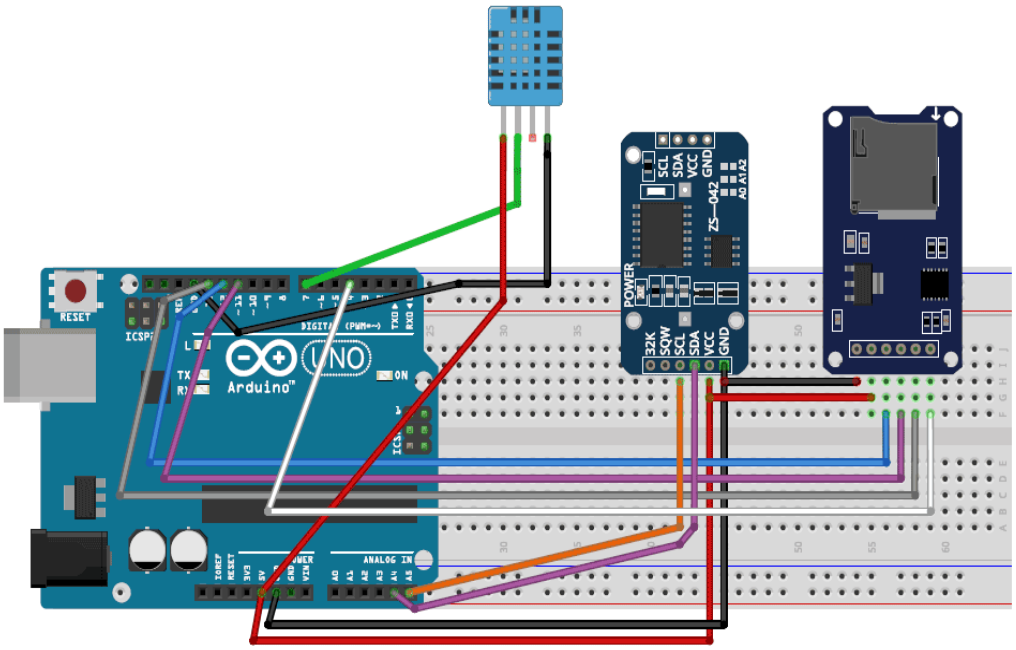

Figure8: Design circuit

As shown in the circuit diagram the connections are straight forward since, we have used them as modules, we can directly build them on a breadboard. The connections are further classified in the table below,

Arduino Pin	Module Pin
Temperature Sensor – DHT11	
V_{cc}	5V
GND	GND
Out	Pin 7
RTC module DS3231	
V_{cc}	5V
GND	GND
SCL	SCL
SDA	SDA

Arduino Pin	Module Pin
SD Card Module	
V_{cc}	5V
GND	GND
MISO	Pin 12
MOSI	Pin 11
SCK	Pin 13
CS	Pin 4

The RTC module DS 3231 is interfaced with Arduino using the I2C communication (SCL, SDA) and the SD card module is interfaced using the SPI Communication (MISO, MOSI, SCK, CS). The pin 4 is defined as the CS pin, we can change them to any other pin if required.

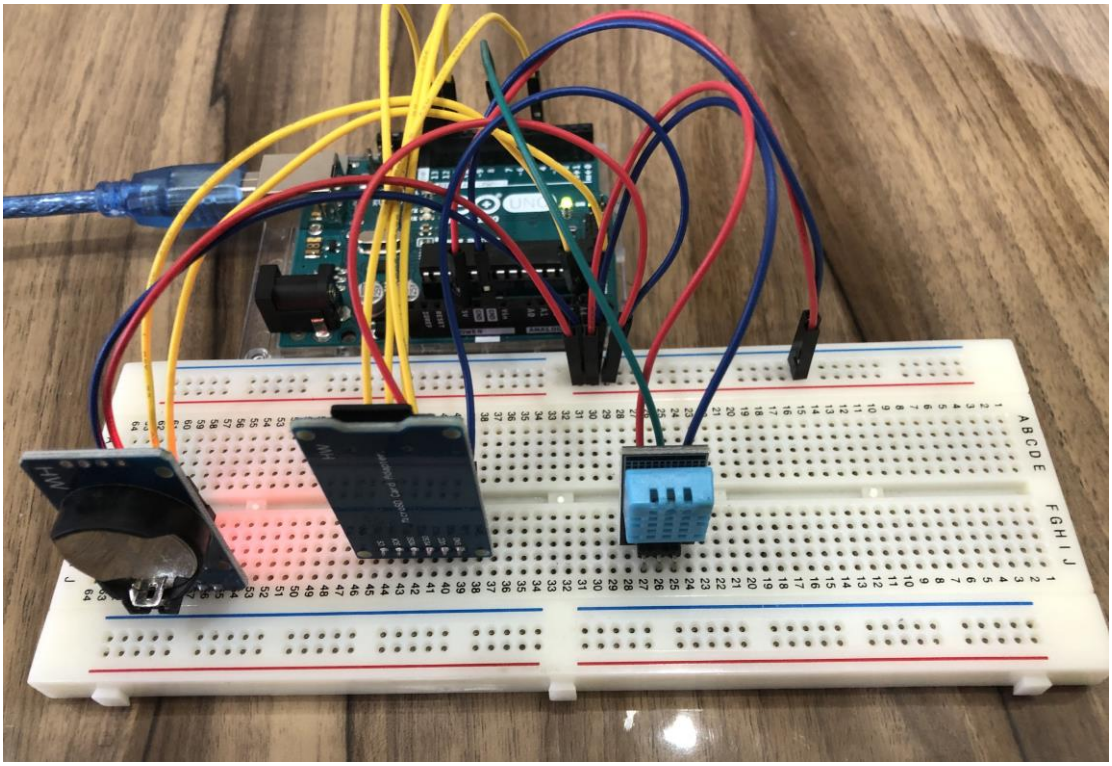

Figure9: Design working circuit

5.0 Code

We must write the Arduino program which can do the following.

- Read data from DTH11 Sensor,
- Initialize the I2C bus to read data from RTC module,
- Initialize the SPI bus to interface the SD card module with Arduino,
- Store the Date, Time, Temperature and Humidity into the SD card,
- Store the Date, Time, Temperature and Humidity on an Excel Sheet running on a computer.

The above steps might sound complicated, but they are quite easy since we have the libraries to do the hard job for us. We have to download the following two libraries

- DHT11 Sensor Library
- DS3231 RTC module library

To feed the data from Arduino lively into an Excel sheet on a computer we shall also need to install software called **PLX-DAQ** provided by Parallax Inc. Download this software from the Moodle and install them based on your operating system. This should have created a folder named **PLX-DAQ** on your **desktop**. We shall use this later in our data capture section.

Now after adding libraries and after installing the software, you can use the **complete sketch** (download from Moodle) and upload the sketch into the Arduino board. The sketch is written simple as possible and the explanations are also given through comment sections.

5.1 Capturing Data from DS3231

DS 3231 is a RTC (Real Time Clock) module. It is used to maintain the date and time for most of the Electronics projects. This module has its own coin cell power supply using which it maintains the date and time even when the main power is removed or

the MCU has gone though a hard reset. So, once we set the date and time in this module it will keep track of it always.

Using this module is quite easy because of the library provided by Arduino.

```
/ Init the DS3231 using the hardware interface
DS3231  rtc(SDA, SCL);
void Initialize_RTC()
{
  // Initialize the rtc object
 rtc.begin();

//the following lines can be uncommented to set the date and time for
the first time

//rtc.setDOW(FRIDAY);     // Set Day-of-Week to SUNDAY
//rtc.setTime(18, 46, 45);     // Set the time to 12:00:00 (24hr format)
//rtc.setDate(6, 30, 2017);   // Set the date to January 1st, 2014

`
```

Note: When using this module for the first time you must set the date and time. It can be done by simply removing the comments as mentioned above and writing the date and time. Make sure you comment them back and upload it, else each time you run the board the date and time will be set again.

5.2 Reading Data from DHT11

DHT11 is a Temperature come Humidity sensor. It sends the values of temperature and humidity as an 8-bit data serially through the output pin of the module. The library reads this data by using the software serial function of the Arduino.

```
#define DHT11_PIN 7 //Sensor output pin is connected to pin 7
dht DHT; //Sensor object named as DHT
void Read_DHT11()
{
int chk = DHT.read11(DHT11_PIN);
}
```

We have connected the output pin to pin 7 as example you can choose any pin that supports software Serial. The ***DHT.read(pin number);*** will read the value of the

temperature and humidity and store it in the parameter ***DHT.temperature*** and ***DHT.Humidity*** respectively.

5.3 Initializing the SC Card Module

```
void Initialize_SDcard()
{

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


   // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  File dataFile = SD.open("LoggerCD.txt", FILE_WRITE);

  // if the file is available, write to it:

  if (dataFile) {
    dataFile.println("Date,Time,Temperature,Humidity"); //Write the first row of the excel file
    dataFile.close();
  }
}
```

Using an SD card with Arduino is easy because of the SD card library which added to the Arduino IDE by default. In the SD card initialize function we have to create a text file named “LoggerCD.txt” and write the first row of our content. Here we separate the values by using a **“,”** as a delimiter. Meaning when a comma is placed it means we have to move to the next cell in the Excel sheet.

5.4 Writing Data to SD Card

```
void Write_SDcard()
{
   // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  File dataFile = SD.open("LoggerCD.txt", FILE_WRITE);

  // if the file is available, write to it:
  if (dataFile) {
    dataFile.print(rtc.getDateStr()); //Store date on SD card
    dataFile.print(","); //Move to next column using a ","

    dataFile.print(rtc.getTimeStr()); //Store date on SD card
    dataFile.print(","); //Move to next column using a ","

    dataFile.print(DHT.temperature); //Store date on SD card
    dataFile.print(","); //Move to next column using a ","

    dataFile.print(DHT.humidity); //Store date on SD card
    dataFile.print(","); //Move to next column using a ","

    dataFile.println(); //End of Row move to next row
    dataFile.close(); //Close the file
  }
  else
  Serial.println("OOPS!! SD card writing failed");
}
```

Also, our intention is to **save the Date, Time, Temperature and Humidity** into the SD card. With the help of the DS 3231 library and the DHT11 library the Arduino can read all these four parameters and storing them into the following parameters as shown in table below

Date	rtc.getDateStr());
Time	rtc.getTimeStr());
Temperature	DHT.temperature
Humidity	DHT.humidity

Now we can directly use these parameters to store the data on the SD card using the print line

```
dataFile.print(parameter);
```

Each parameter is separated by a comma to make it look legible and a ***dataFile.println()****;* is used to indicate the end of the line.

5.5 Writing Data to PLX-DAQ

PLX-DAQ is a Microsoft Excel Plug-in software that helps us to write values from Arduino to directly into an Excel file on our PC.

To use this software with Arduino we must send the data serially in a specific pattern just like displaying value on serial monitor. The key lines are explained below,

```
void Initialize_PlxDaq()
{
Serial.println("CLEARDATA"); //clears up any data left from previous projects
Serial.println("LABEL,Date,Time,Temperature,Humidity"); //always write LABEL, to indicate it as
first line
}
void Write_PlxDaq()
  {
    Serial.print("DATA"); //always write "DATA" to Inidicate the following as Data
    Serial.print(","); //Move to next column using a ","

    Serial.print("DATE"); //Store date on Excel
    Serial.print(","); //Move to next column using a ","

    Serial.print("TIME"); //Store date on Excel
    Serial.print(","); //Move to next column using a ","

    Serial.print(DHT.temperature); //Store date on Excel
    Serial.print(","); //Move to next column using a ","

    Serial.print(DHT.humidity); //Store date on Excel
    Serial.print(","); //Move to next column using a ","

    Serial.println(); //End of Row move to next row

  }
```

The software can recognize keywords like LABEL, DATA, TIME, DATE etc. As shown in the Initialize function the keyword “LABEL” is used to write the first ROW of the Excel sheet. Later in the Write function we use the keyword “DATA” to indicate that the following information should be considered as DATA. To indicate that we have to move to next row we have to use comma **(“,”)**. To indicate the end of row we have to send a ***Serial.println();.***

As said earlier we can write the system date and time by sending the keywords “DATE” and “TIME” respectively as shown above.

Note: Do not use serial monitor when using this PLX_DAQ software or wise verse.

6.0 Data Capturing

Once the hardware and the sketch are ready it is time to burn the program into the Arduino Board. As soon our program gets uploaded, the temperature and humidity values will start to get stored in your SD card. We have to follow the steps below to enable PLX-DAQ to log the into Excel sheet in the computer.

Step 1: Open the “Plx-Daq Spreadsheet” file that was created on your desktop during installation.

Step 2: If there is a Security block, click on *Options->Enable the content -> Finish -> OK* to get the following screen.

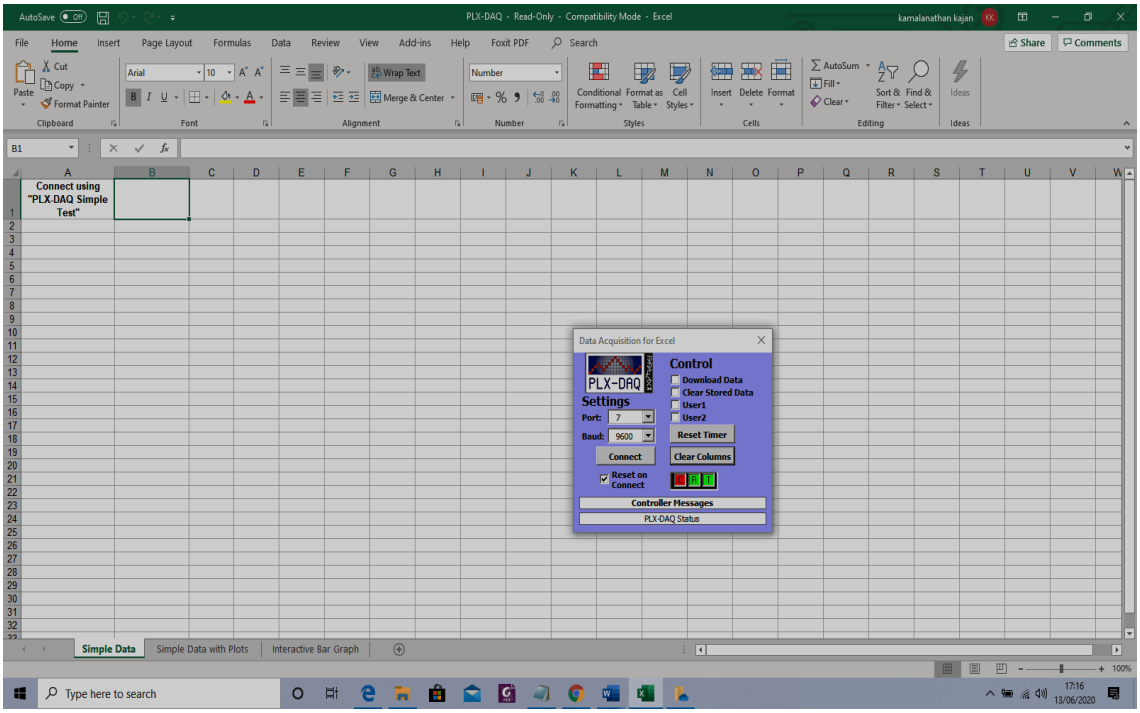

Figure10: Excel work sheet with Plx-Daq

Step 3: Now select the baud rate as “9600” and the port to which your Arduino is connected and click on Connect. Your values should start to get logged like shown in the picture below

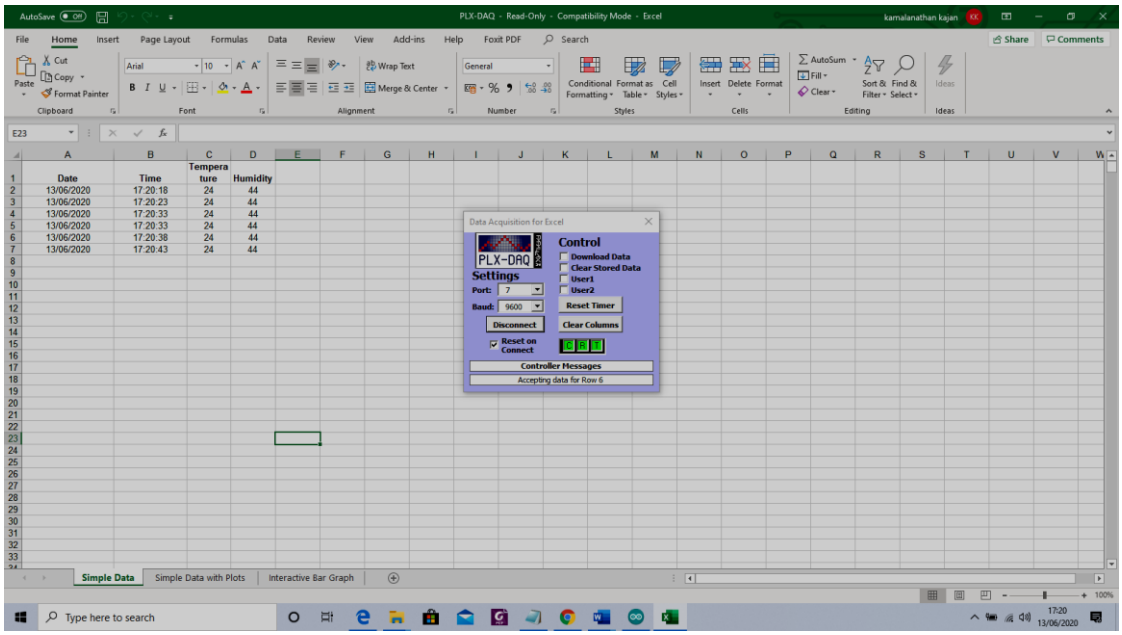

Figure11: Excel work sheet data capturing using with Plx-Daq

We can leave this excel sheet open and monitor the values as they get logged. As this is happening the SD card would also start saving the same data. To check the SD card data, simply remove the SD card and retrieve the data in a computer. You should find a text file named *"LoggerCD.txt"* in it. When opened it would look something like this.

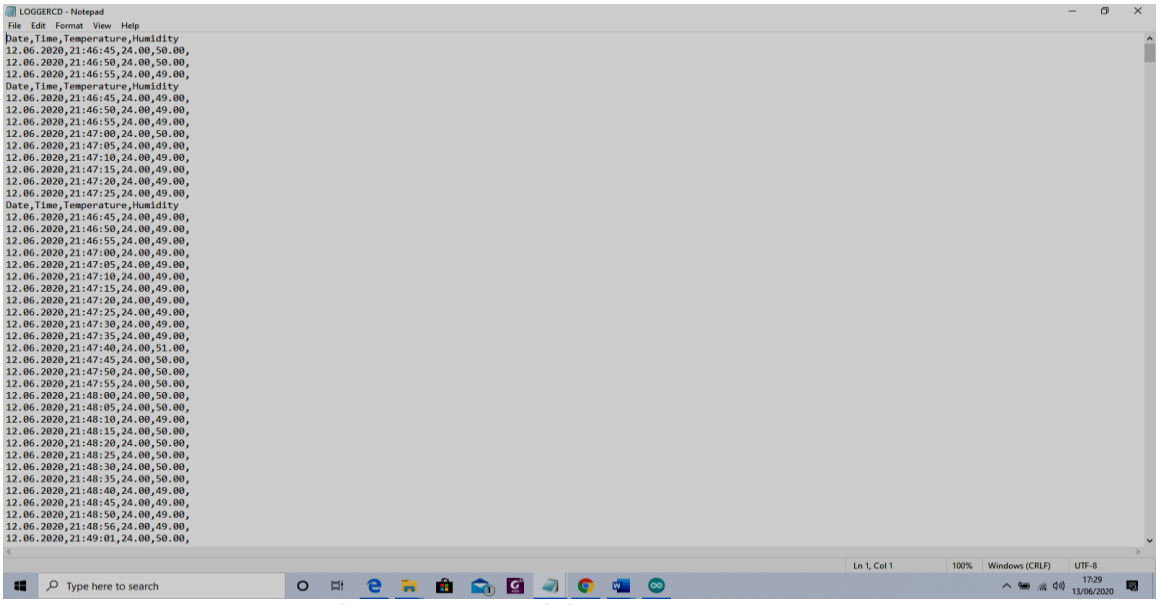

Figure12: SD card data capture

7.0 Tasks

1. Understand the overall concept of this design,
2. Familiarise with the hardware setup,
3. Understand the given sketch for this design,
4. When you are capturing the data (using the given sketch) in the Excel, your time is not showing correctly. Try to find a way to correct this issue,
5. Remove the micro SD card from the module and compare the SD card data with Excel data,
6. Try to increase / decrease the temperature / humidity and observe the results in the Excel. Can you observe the changes according to your input adjustments?
7. Suggestions for future improvements and ideas to implement this design in the real world.

8.0 Sample MCQs for Quiz

8.1) Identify this component.

1) Wind sensor
2) Ultrasonic sensor
3) Speaker
4) RTC module

8.2) Select the correct ways of defining a function

1) def myfunction()
2) int myfunction (float)
3) void myfunction ()
4) void myfunction (int,int,int)

8.3) What is the communication protocol used by SD (Secure Digital) cards?

1) I2Cwrong
2) UART
3) SPI
4) USART

8.4) An Arduino Uno is **most** suited for _____.

1) Controlling peripherals
2) Powering peripherals
3) Controlling and powering peripherals
4) None of the above

End of Lab - 6