ECE 480

Fall, 2015

SD Cards and Data logging with Arduino

Application Note

Xue Cheng

11/13/2015

Executive Summary

This Application Note is written for new users to Arduino, and it is intended to help users to get experience with basic PLC control of the Arduino microcontroller. Furthermore, this application note will provide guideline on how to connect a SD card to the Arduino board and achieve data logging process to record data from desired sensors.

Keyword

Arduino Uno

Arduino Software (IDE)

Sensor Reading

SD.h Arduino library

Data Logging

SD Card

Introduction

Nowadays, digital control systems are widely used in all industries and everyday life. Sensor data acquisition and association are critical parts for digital control system. Data recording is important for tracking the performance of the sensors and also improving the digital control system. A safety system, for the team project related to this application note, requires data recording to report any near-miss accidents' information and also keeps records of possible risk factors to help the industry to improve their safety in work environment. The recorded data should be easy to read for anyone with or without technical knowledge in microcontroller. Therefore, the data should be recorded within a SD card or USB and in a form such as excel for easy access and understanding. It is also important to refresh the memory card to enable for new data recording once a while.

Objective

In this application note, readers with or without knowledge of microcontroller should be able to understand the process of basic data logging with Arduino Uno using SD card. Basic instruction of setting up and operating an Arduino Uno chip will be introduced in this notes. Also this application note will provide information on how to convert the signal from sensors to a readable data form presented in Excel.

Issue

During the team project, team member need to design a safety system that can create a safety cone around an operating crane hook and warn any person within a dangerous zone. Team member uses a thermal sensor to create the cone of safety, and if anyone accidentally enters the zone, signal will be send to the microcontroller then an output signal will trigger the alarm. In this project, sensors need to be controlled by microcontroller and also data at each time that alarm is triggered need to be recorded for future analyze and possible improvement to the safety system. Data that's been recorded need to be easy to access and in a form that's easy to understand and analyze for anyone. Also the memory storage of the data need to be able to reset after a certain time period to enable enough space to write new data in.

Hardware

In order to accomplish a working system with operating sensors and data logging task, users first need to get an Arduino Uno board (Figure 1) along with a standard USB cable (Figure 2).

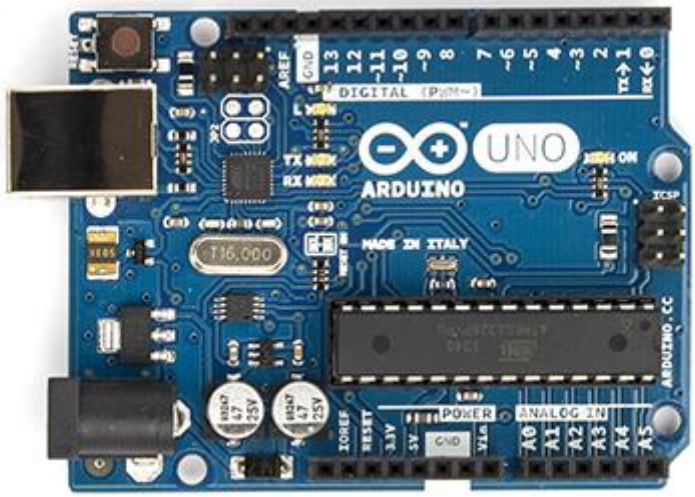

Figure 1. Arduino Uno Board

Arduino is an open-source prototyping, it simplifies the amount of hardware and software development you need to do in order to get a system running. The Arduino boards are able to read inputs such like temperature measured with a sensor, distance between two points, or a Twitter message and then turn it into an output such like activating a motor, turning on an LED, publishing something online.

On the hardware side, Arduino platform already has the power and reset circuitry setup as well as circuitry to program and communicate with the microcontroller over USB. In addition,

the I/O pins of the microcontroller are typically already fed out to sockets for easy access. Datasheet is clear enough to find the correct pin for connection.

On the software side, Arduino provides its own libraries with functions and free codes that make programming the microcontroller easier and save time from building supporting circuity and writing low level codes. The Arduino software which is called IDE provides libraries with functions that keep us from writing with C from very low level. Also the functions provided by Arduino is easy to understand. The simplest of these are functions to control and read the I/O pins rather than having to fiddle with the bus/bit masks normally used to interface with the Arduino mega I/O. More useful are things such as being able to set I/O pins to PWM at a certain duty cycle using a single command or doing Serial communication.

Figure 2. Standard USB cable

USB cable will be used to program the Arduino MEGA ADK board and supply power to the board.

Users will also need a breadboard (Figure 3) which is important for prototyping the designed circuit

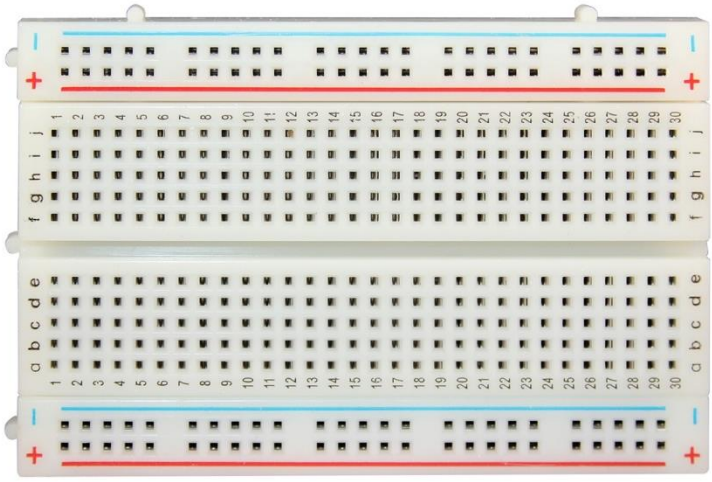

Figure 3. Breadboard

In order to test the performance of the controlled circuit, users can use resistors along with LED to analyze the output of the circuit time by time.

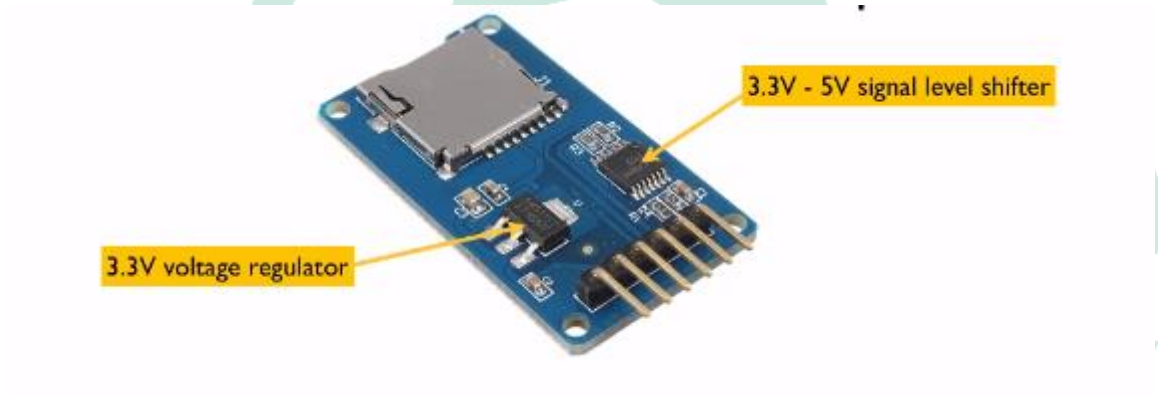

Figure 4. Catalex Arduino Micro SD Adapter

Adapter combines a SD card slot with a 3.3V – 5V level shifter and a 3.3V voltage regulator. This enables direct hookup to the Arduino's SPI pins.

Figure 5. Micro SD card and Adapter

Along with the Catalex Arduino Micro SD adapter, a micro SD card and Adapter for it will needed as a memory storage to record data and read the data.

SD card need to be formatted into the FAT format before use. A 2 GB or less card can be formatted in FAT or FAT32. Cards larger than 2 GB should be formatted in FAT32. Cards larger than 32 GB should be formatted in exFAT. Formatting SD card into FAT on Windows system can be done following the given instructions below:

1. Place the SD card into the SD card slot in the device
2. Press the Windows key + E on the keyboard to open Windows Explorer/File Explorer
3. Click This PC (Windows 10 only)
4. Locate the Removable Disk icon representing the SD card
5. Right-click on the Removable Disk icon
6. Click on Format
7. Change Allocation unit size to Default allocation size
8. Ensure Quick Format is checked
9. Click Start

Follow the datasheet and pin map of Arduino Uno, users should be able to connect the SD card reader to the Arduino Uno along with the sensor.

Pins that need to be connected are given in Figure 7.

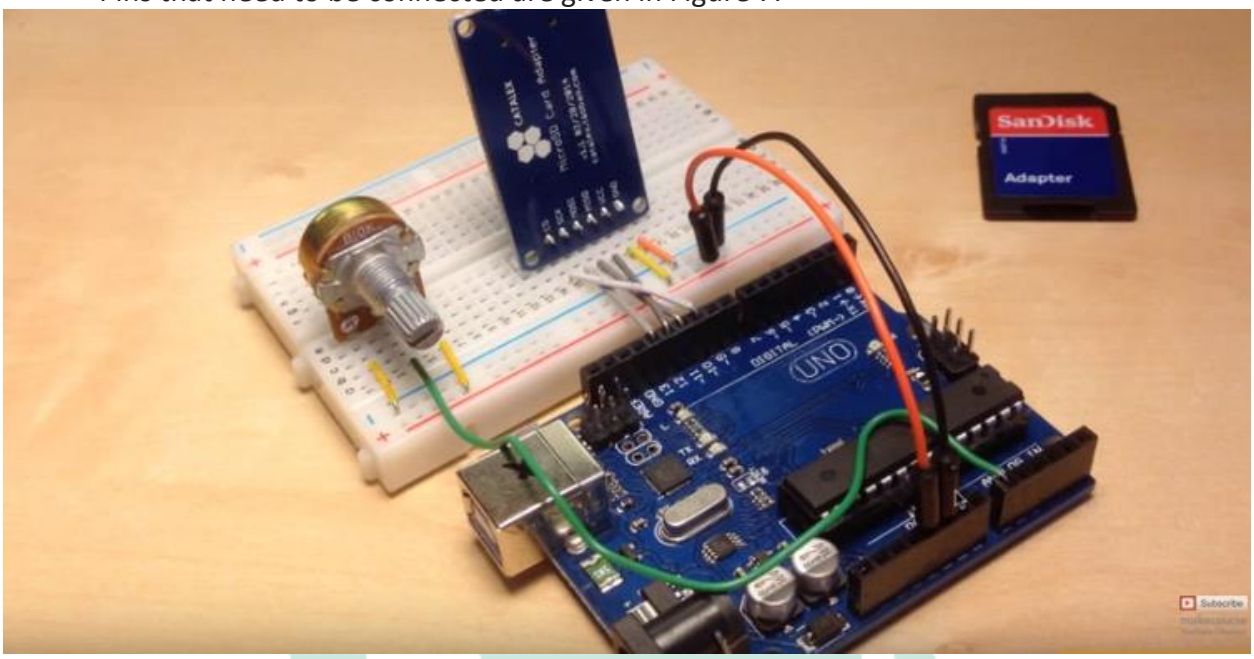

Figure 6. Connecting SD Card Reader

Connecting the SD Card Reader

Sd Card Reader Pin	Arduino Pin	Details
GND	GND	Common Ground
3.3 V – (NOT USED)		
+5	5V	Power
CS	4	Chip Select
MOSI	11	SPI Data
SCK	13	Clock
MISO	12	SPI Data
GND	GND	Common Ground

Figure 7. Connecting SD Card to Arduino Uno

Software

After connected the hardware, users now should start the software and coding for the data logging purpose.

First of all, download the Arduino software (IDE) from Arduino official website for code composing.

Code

For SD data logging, it is important to include SD.h library. The SD library allows for reading from and writing to SD cards. It supports FAT16 and FAT32 file systems on standard SD cards and SDHC cards.

The communication between the microcontroller and the SD card uses SPI, which takes place on digital pins 11, 12, and 13 on Arduino Uno Board. Additionally, another pin must be used to select the SD card. In this tutorial we will use hardware SS pin - pin 10 on Arduino Uno. Note that even if you don't use the hardware SS pin, it must be left as an output or the SD library won't work.

```
#include <SD.h> //Load SD card library

#include<SPI.h> //Load SPI Library
```

Then we will include Wire.h library for I2C communication. This library can be found and downloaded online for free.

```
#include "Wire.h"
```

Define variable for temperature data.

```
float tempC;  // Variable for holding temp in C

float tempF;  // Variable for holding temp in F
```

Set the chipSelect pin for the SD card Reader

```
const int chipSelect = 4;
```

The following loop is a set-up function that will be used to check if the SD card is present and initialize the SD card before future processing the data. This function will only happen once when you start the program. If the card is not present, it will show that there's an error with the SD card and will stop the process to the future steps.

Also, in the setup(), call SD.begin(), naming pin 4 in this case as the CS pin. This pin varies depending on the make of shield or board.

```
void setup() {
 // Open serial communications and wait for port to open:
 Serial.begin(9600);
 while (!Serial) {
  ; // wait for serial port to connect. Needed for native USB port only
 }


 Serial.print("Initializing SD card...");

 // see if the card is present and can be initialized:
 if (!SD.begin(4)) {
  Serial.println("Card failed, or not present");
  // don't do anything more:
  return;
 }
 Serial.println("card initialized.");
```

At last, create a new file with SD.open() named "TempData.txt". FILE_WRITE enables read and write access to the file, starting at the end. If a file "TempData.txt" was already on the card, that file would be opened.

```
mySensorData = SD.open("TempData.txt", FILE_WRITE);

}
```

If the set-up loop initialized the SD card successfully, the serial screen will display card initialized then process to the main function loop as shown below. In this loop, the code will build and open a txt file on the SD card to store data that's read from the sensors.

First of all, user should define the data that's read from the thermal sensor which is used in this project. The data should be recorded in Celsius according to the data sheet. Users can transfer this temperature data into Fahrenheit for convenience.

```
void loop() {

tempC = mySensor.readTemperature(); //  Read Temperature from BMP180

tempF = tempC*1.8 + 32.; // Convert degrees C to F
```

Use the following command to display the data that's read from the sensor on the serial monitor given in Arduino IDE software. The given format below will make the data easier for users to understand.

```
if (mySensorData) {
Serial.print("The Temp is: "); //Print Your results
Serial.print(tempF);
Serial.println(" degrees F");
```

Then use myFile.println() to write a string to the card, followed by a carriage return. Once the content is written, close the file.

```
mySensorData.print(tempC);                    //write temperature data to card
mySensorData.print(",");                      //write a comma
mySensorData.print(tempF);                    //write temperature data to card
mySensorData.close();                         //close the file
```

After edit the command code, check and verify the code with IDE make sure there's no error in the code then save it. It there's any error, it should show on the bottom of the command window, check the error and edit the code before future process. After the code is checked and saved, user may connect the hardware (Figure 6.) to the PC that contains the written code with a USB cable (Figure 2). Use IDE software to upload the code to Arduino Uno. Open the Serial screen to monitor the performance of the Arduino.

Expected Results

After compile and run the code to Arduino, the code will first check if the SD card is present and functioning. User should expect the selected sensor to write data into the selected port on Arduino board under certain circumstance. The code should create a txt file on the SD card. Remove the micro SD card and read it with SD adapter on PC, it should be easy to transfer the data into excel format by “copy” and “paste”. By converting the data into excel, user can generate more visualized plot with the recorded data.

Recommendation

Future improvement can be done to this design by inserting a RTC(Real-Time-Clock) along with the design, this will provide more detailed information of the data such as the date and time of the recorded data. The implementation of RTC will provide useful information to better analyze recorded data.

Conclusion

This application note guided the user how to setup SD card data logging with Arduino Uno following a step-by-step procedure. The tutorial also introduced how to format SD card and use the SD.h library provided by Arduino. During the software section, it briefly explained the command line and method to edit, compile and run a file. Through this application note, user should have experience with some of the Arduino libraries. Further application of this data logging method with other kind of sensors may use different library. A common sensor usually has a written library online which is available for download. Overall, this application note provides a basic method for data logging on SD card, further modification can be done to record more detailed information.

Reference

1. Arduino.cc,. 'Arduino - Datalogger'. N.p., 2015. Web. 23 Nov. 2015. https://www.arduino.cc/en/Tutorial/Datalogger

2. "Technology Tutorials." *Technology Tutorials*. N.p., n.d. Web. 23 Nov. 2015. http://www.toptechboy.com/arduino/arduino-lesson-21-log-sensor-data-to-an-sd-card/