Lab 5 Appendix

Book Title: *Real-Time Environmental Monitoring: Sensors and Systems, Second Edition – Lab Manual*

Author: Miguel F. Acevedo (https://orcid.org/0000-0002-5136-3624)

Using flash memory for datalogging with Arduino

This section was part of Chapter 6 of the First Edition. This flash memory is no longer available but the example illustrates the use of UART when interfacing with Arduino.

The Arduino flash memory (32KB) is not enough for many datalogging applications where we need to store larger amount of data but not as much as provided by many available flash-memory devices (say 32GB). However, flash memory add-ons are available to provide a moderate amount of memory. For example, the Atlas Scientific ENV-32X Embedded Data Logging Subsystem can hold 32 MB (Atlas Scientific 2014).

Read/Write/Erase operations are done using simple one-character commands in asynchronous serial format. We can write a 128-byte character strings (convenient size for monitoring applications) to memory in just 3 milliseconds. It is then possible using read commands to retrieve the data from the flash drive from a PC or another computer. The ENV-32X Embedded Data Logger requires no additional hardware and no drivers to operate. Simply output the string to be stored to the 32X Embedded Data Logger. It can hold hundreds to thousands of these short data strings that are commonly used when logging sensor data.

Other features of the EMV-32X include: two read back methods, simple asynchronous serial connectivity, simple instruction set consisting of only seven commands, two debugging LEDs, 3.3 volt operation, low power consumption, and very low weight and footprint.

The seven commands are L1, L0, D, R[0,1], E, M, and I.

- L1: enable both debugging LEDs
- L0: disable LEDs
- D: delete all stored data,
- R: read, followed by 0 (R0) output data as CSV files; followed by 1 (R1) outputs data page by page.
- E: stops data read
- M: return memory space available
- I: report version number

The EMV-32X has two pins VCC and GND that are connected to the 3.3 V and GND pins of the Arduino. Also has TX and RX pins that are connected to RX and TX pins of the Arduino, which are pins 0 and 1 respectively). The EMV-32 green LEDs light up when transmitting and the red LEDs light up when receiving data (Figure 5.1).

As an example, students in an environmental monitoring class experimented reading the output of a EC-5 soil moisture sensor every 15 min, store that data into the datalogger, and retrieve by using the R0 command. Here the Arduino is powered with a 9V battery connected to the Vin pin and GND. The code used is

```
#include <Time.h>
#include <SoftwareSerial.h>
#define rxpin 1
```

```
#define txpin 0
SoftwareSerial mySerial(rxpin,txpin);  //enable soft serial port
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
int analogpin = 0;         // value read from the pot
int initString = 0;
String inputString="";
String sensorString="";
int varx = 0;
String dataString="";
#define TIME_MSG_LEN  11   // time sync to PC is HEADER followed by Unix time_t as ten ASCII
digits
#define TIME_HEADER  'T'   // Header tag for serial time sync message
#define TIME_REQUEST  7    // ASCII bell character requests a time sync message

void setup()
{
Serial.begin(38400);
mySerial.begin(38400);
inputString.reserve(30);
sensorString.reserve(60);
//setTime(hr,min,sec,day,month,yr);
setTime(5,11,0,4,12,13);
delay(3000);
}

void loop()
{
initString=analogRead(analogInPin);
if (varx == 0)
{
varx++;
delay(2000);
dataString += "Time";
dataString += ",";
dataString += "Volumetric water content";
Serial.print(dataString);
mySerial.print(dataString);
}
delay(1000);
// make a string for assembling the data to log:
        inputString += String(month());
        inputString += "/";
        inputString += String(day());
        inputString += "/";
        inputString += String(year());
        inputString += "-";
        inputString += String(hour());
        inputString += ":";
        inputString += String(minute());
        inputString += ":";
        inputString += String(second());
        inputString += ",";
        inputString += " ";
        inputString  += String(initString);
        inputString +="\n";
        mySerial.print(inputString);
        Serial.print(inputString);
        inputString="";
      char inchar=(char)Serial.read();
      data(inchar);
        }
void digitalClockDisplay(){}
void printDigits(int digits){
// utility function for digital clock display: prints preceding colon and leading 0
  Serial.print(":");
for(i=0;i<10000;i=i+15)
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}
    Voiddata(char data)
{
     if(data=”R0”)
     {
           get(mySerial.read);
           SensorString+=data;
}
         Serial.print("sensorString is");
         Serial.print(sensorString);
         sensorString="";

}
```

Which produces a csv file containing date, time, and value read,

```
12/5/2013,4:12:10,83
12/5/2013,4:27:10,90
12/5/2013,4:42:10,105
12/5/2013,4:57:10,112
12/5/2013,5:12:10,123
12/5/2013,5:27:10,135
12/5/2013,5:42:10,147
12/5/2013,5:57:10,153
12/5/2013,6:12:10,158
12/5/2013,6:27:10,162
```

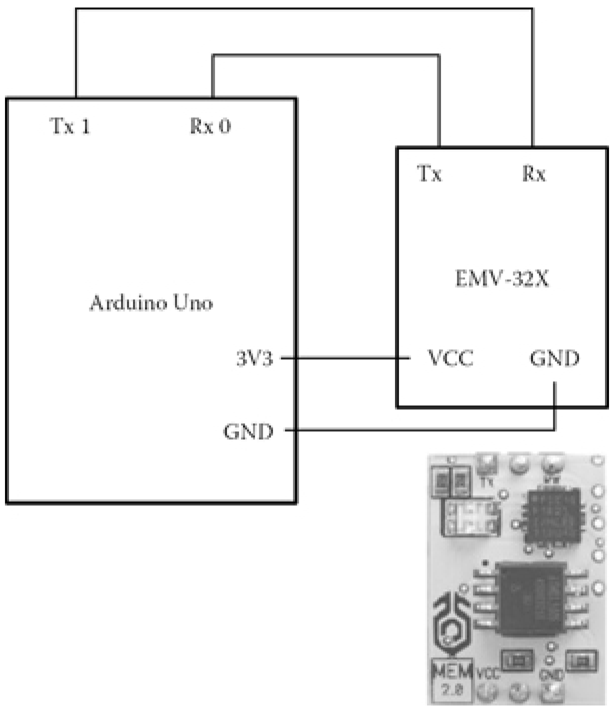

Figure 5.1. ENV-32X flash memory for datalogging with Arduino. (From Atlas Scientific, *ENV-32X Flash Memory Data Logger*, 2014, Retrieved October 2014, from www.atlas-scientific.com/product_pages/embedded/env-32x.html.)

References

Atlas Scientific. 2014. *ENV-32X Flash Memory Data Logger*. Atlas Scientific accessed October. https://www.atlas-scientific.com/product_pages/embedded/env-32x.html.