VIRTUAL COURSE
BUILD YOUR OWN DATA LOGGER

WILDLABS.NET
[ The conservation technology network ]

FREAKLABS
MODULE 4
TYING IT ALL TOGETHER
Recap - Basic Function Blocks

- Command line
- Temperature and Humidity sensor
- Analog to Digital Converter
- Real Time Clock
- SD Card
- Interrupts
Good Programming Practice

• Comments
• No ‘magic’ numbers
• Try to name your variables intuitively
• No massive blocks of code (try not to)
Use Comments

• Line comments eg. // this is a line comment

• Multi-Line / block comments eg. /* this is a block comment that spans multiple lines */
No ‘Magic Numbers’

• Leave a calculation trail of how you came to a number so you remember, and others can easily understand

• Use define constants so numbers are changed in one area only eg.

```c
9 // ADC
10 #define ADC_REF_VOLTAGE 3.3
11 #define ADC_LEVELS 1024
12 #define ADC_SCALE_FACTOR 2
```

• Explain calculations with well named variables eg

```c
float getBatteryVoltage()
{
    int battAdc = analogRead(pinBatt);
    float voltsPerAdcUnit = ADC_REF_VOLTAGE / ADC_LEVELS; // this is the voltage per ADC unit
    float pinVoltage = battAdc * voltsPerAdcUnit;
    float battVoltage = pinVoltage * ADC_SCALE_FACTOR; // this is the voltage at the battery
    return battVoltage;
}
```
Use Intuitive Names for Variables

- Use variables names that give context to what’s happening eg.
  \( a == \text{high} \) versus \( \text{pinPIR} == \text{high} \)

- Use camelcase or underscores to delineate abbreviations or words in variables eg.
  \( \text{flagMotionEvent} \) or \( \text{flag}\_\text{motion}\_\text{event} \)

```plaintext
if (a == HIGH)
{
    b = 1;
}
else
{
    b = 0;
}
if (pinPIR == HIGH)
{
    flagMotionEvent = 1;
}
else
{
    flagMotionEvent = 0;
}
```
No Massive Blocks of Code or Huge Single Functions

- Group code into functional blocks, and keep functions separate
- Easier to read and maintain

```c
// handle logging data
do {  
  int battAdc = analogRead(pinBatt);  
  float voltsPerAdcUnit = ADC_REF_VOLTAGE / ADC_LEVELS;  
  float pinVoltage = battAdc * voltsPerAdcUnit;  
  float battVoltage = pinVoltage * ADC_SCALE_FACTOR;  
  return battVoltage;
} while (true);
```
Data Logger Block Diagram

**Setup**
- Initialize Serial
- Initialize sensor
- Initialize SD card
- Initialize timer
- Initialize interrupts
- Sleep

**Periodic Logging**
- Sleep
  - Timer Interrupt
  - Realtime Clock
  - Battery (ADC)
  - Temp Humidity Sensor
  - Append "RTC"
  - SD Card
  - SD Card

**Motion Logging**
- Sleep
  - PIR Interrupt
  - Realtime Clock
  - Battery (ADC)
  - Temp Humidity Sensor
  - Append "PIR"
  - SD Card
  - SD Card

**Button Pressed**
- Sleep
  - Button Interrupt
  - Realtime Clock
  - Battery (ADC)
  - Temp Humidity Sensor
  - Append "BTN"
  - SD Card
  - SD Card
Breaking Down Main Functions

void setup()
Initialization

void loop()
look for interrupt flags

void logData(int source)
collect data and write to card

float getBattery()
returns battery voltage

void cmdFileRead()
void cmdFileList()
void cmdFileDelete()
Testing purposes only

void sdDateTime
Add date / time to file creation

void isrButton()
Button ISR

void isrMotion()
Motion ISR

void isrTimer()
Timer ISR
Set Up

- PIR Motion sensor interface
- DHT11 sensor interface
- microSD card slot
- Push Button interrupt
Code #1 – Includes & Defines

```cpp
#include <cmdArduino.h>
#include <SdFat.h>
#include <Rtc_Pcf8563.h>
#include <DHT.h>

// logfile
#define LOGFILE "LOGFILE.TXT"

// ADC
#define ADC_REF_VOLTAGE 3.3
#define ADC_LEVELS 1024
#define ADC_SCALE_FACTOR 2

// pushbutton
#define DEBOUNCE_TIME 200

// sensor
#define DHTTYPE DHT11
#define DHTPIN A0

// timer intervals
#define TIMER_MINUTE 1
#define TIMER_HOUR 60

// interrupt sources
#define TIMER 0
#define MOTION 1
#define BUTTON 2
```
Code #2 – Create Objects

```c
// create objects
SdfFat sd;
File myFile;
DHT dht(DHTPIN, DHTTYPE);
Rtc_Pcf8563 rtc;

// create and set pins
int pinSD = 10;
int pinLED = 4;
int pinBatt = A6;

// create and set interrupts
int intpNumRtc = 0;
int intpNumPIR = 1;
int intpButton = 2;

// create flags
volatile int flagTimer = 0;
volatile int flagPIR = 0;
volatile int flagButton = 0;
```

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Interrupt Number</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Time Clock</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>PIR Motion Sensor/Aux Intp</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Pushbutton 0</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>
Code #3 – void setup ()

```c
void setup()
{
    // init LED
    pinMode(pinLED, OUTPUT);
    digitalWrite(pinLED, LOW);
    mcu.begin(57600);
    Serial.println("Module 4-0: Tying It All Together");

    // for testing only
    cmd.add("fread", cmdFileRead);
    cmd.add("ls", cmdFileList);
    cmd.add("rm", cmdFileDelete);

    // initialize peripherals that need it
    sd.begin(pinSD);
    dht.begin();

    // initialize interrupts
    attachInterrupt(intpNumRtc, isrRtc, FALLING);
    attachInterrupt(intpNumPIR, isrPIR, RISING);
    attachInterrupt(intpButton, isrButton, FALLING);

    // misc
    myFile.dateTimeCallback(sdDateTime);
    rtc.setTimer(TIMER_MINUTE, TMR_1MIN, false);
}
```
Code #4 – loop() & Interrupt Event Handlers

```c
void loop()
{
    cmd.poll();

    // handle timer interrupt
    if (flagTimer == 1)
    {
        flagTimer = 0;
        logData(TIMER);
        Serial.println("Timer interrupt received.");
    }

    // handle PIR interrupt
    if (flagFIR == 1)
    {
        flagFIR = 0;
        logData(MOTION);
        Serial.println("Motion interrupt received.");
    }

    // handle button interrupt
    if (flagButton == 1)
    {
        // debounce button
        delay(DEBOUNCE_TIME);
        flagButton = 0;
        logData(BUTTON);
        Serial.println("Button interrupt received.");
    }
}
```
Code #5 – Helper Functions – Logging Data, Battery Voltage

```c
void logData(int type)
{  
    if (type == TIMER)
        myfile.print(" ,RTC");
    else if (type == MOTION)
        myfile.print(" ,PIN");
    else if (type == BUTTON)
        myfile.print(" ,BTN");
    myfile.close();
    Serial.println("Data logged to SD Card.");
}

float getBatteryVoltage()
{  
    int battADC = analogRead(pinBatt);  
    float voltsPerADChbit = ADC_REF_VOLTAGE / ADC_LEVELS;  // this is the voltage per ADC unit
    float pinVoltage = battADC * voltsPerADChbit;  // this is the voltage at the pin
    float battVoltage = pinVoltage * ADC_SCALE_FACTOR;  // this is the voltage at the battery
    return battVoltage;
}
```
Code #6 - Interrupt Service Routines

// ISR for RTC
void isrRtc()
{
    flagTimer = 1;
}

// ISR for PIR motion sensor
void isrPIR()
{
    flagPIR = 1;
}

// ISR for pushbutton
void isrButton()
{
    flagButton = 1;
}
Code #7 – Command Functions

```c
168  //= Command functions
169  //=
170  void cmdFileRead(int argCnt, char **args)
171  {
172      myfile = sd.open(args[1], O_READ);
173      while(myfile.available() > 0)
174      {
175          char c = myfile.read();
176          Serial.write(c);
177      }
178      myfile.close();
179  }
180  }
181  void cmdFileList(int argCnt, char **args)
182  {
183      sd.ls(LS_DATE);
184  }
185  }
186  void cmdFileDelete(int argCnt, char **args)
187  {
188      sd.remove(args[1]);
189      Serial.print(args[1]);
190      Serial.println(" deleted.");
191  }
192  ```
Code #8 – File Creation Date

```c
void sdDateTime(unsigned int *date, unsigned int *time)
{
    rtc.getDateTime();

    // return date using FAT_DATE macro to format fields
    *date = FAT_DATE(2000+rtc.getYear(), rtc.getMonth(), rtc.getDay());

    // return time using FAT_TIME macro to format fields
    *time = FAT_TIME(rtc.getHour(), rtc.getMinute(), rtc.getSecond());
}
```
Code– Compiled

Module 4-0: Tying It All Together
Data logged to SD card.
Motion interrupt received.
Data logged to SD card.
Button interrupt received.
Data logged to SD card.
Button interrupt received.
\texttt{fread LOGFILE.TXT
02/08/2021,10:36:44,30.20,2.00,0.00,PIR
02/08/2021,10:36:48,29.90,2.00,0.00,BTN
02/08/2021,10:36:49,29.90,2.00,0.00,BTN
02/08/2021,10:37:00,29.80,2.00,0.00,RTC
02/08/2021,10:37:21,29.50,1.00,0.00,PIR
02/08/2021,10:37:23,29.00,2.00,0.00,BTN
02/08/2021,10:37:24,29.00,2.00,0.00,BTN
*************** CMD ***************
CMD >>
Code– Compiled

```
LOGFILE.TXT deleted.

*********** CMD ***********
CMD >> ls

*********** CMD ***********
CMD >> Data logged to SD card.
Motion interrupt received.
Data logged to SD card.
Button interrupt received.
Data logged to SD card.
Button interrupt received.
fread LOGFILE.TXT
02/08/2021, 10:37:45, 29.00, 2.00, 0.00, PIR
02/08/2021, 10:37:48, 28.70, 3.00, 0.00, BTN
02/08/2021, 10:37:49, 28.70, 3.00, 0.00, BTN

*********** CMD ***********
CMD >>
```
COMING UP
Module 4.1: Safety Checks and Diagnostics