Arduino Programming

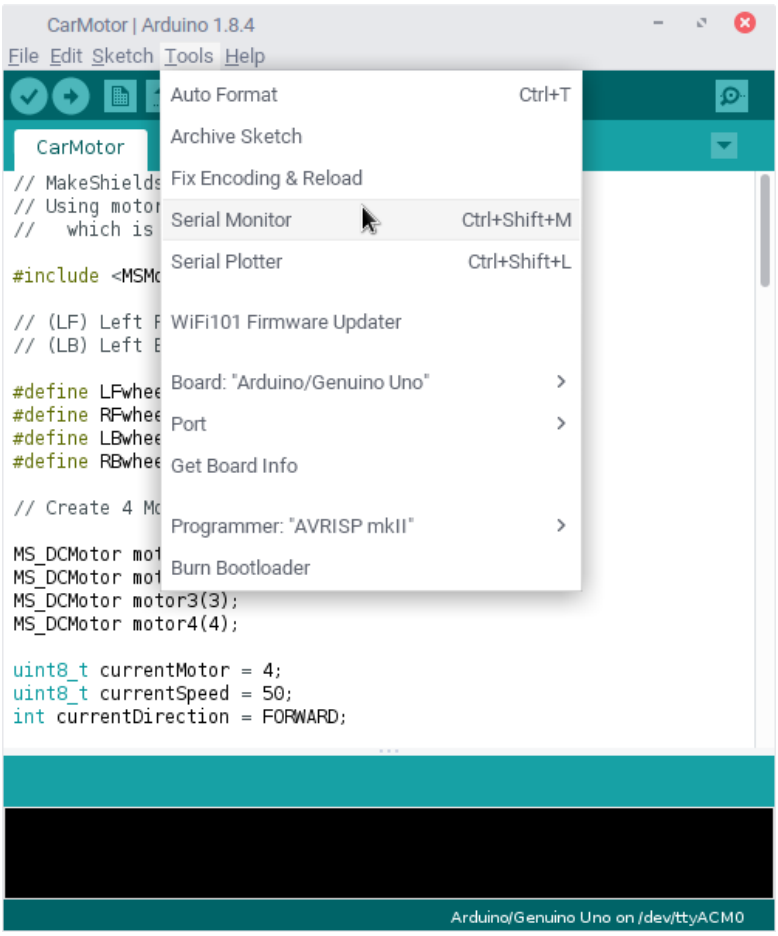
Developed by Mark Webster, Connor Sheeran, Ousema Zayati ???

Summary
In this workshop learn to program a microcontroller such as the Arduino using the Arduino IDE and the C computer language.

Introduction

The Arduino IDE (downloaded from: https://www.arduino.cc/en/main/software) is a development environment that makes it easy to create programs (called sketches) to run on the Arduino family of microcontrollers, and several other microcontroller boards.

The Arduino IDE is a cross platform, free software development system for programming microcontrollers. It has a code editing window, a board manager, a library manager, a debug message window, and built in tools to compile and download the code to a microcontroller connected by a USB cable. The Arduino IDE also provides a set of objects such as Serial for interacting with the host's computer. There are debugging helps for viewing serial output (Serial Monitor) and a graphical logging window (Serial Plotter)



The Arudino IDE hides the C main() function and only requires a programmer to fill in a setup() and a loop() function.

The setup() function is called once when the program starts, the loop() function is called repeatedly after the setup() is done.

C Basics

All computer languages offers four features:

1) A way to give a name to a part of memory and to assign or retrieve data from that memory. Those labels are called variables.
2) Branches in the program flow based on some logical condition. In C these branches are “if” and “switch”
3) Looping, In C these are “for”, “while”,”do while”
4) A way to group and reuse code. C uses functions and structs (C++ has objects and methods)

Variables

C is strongly typed so what data the variable contains must be declared. Common types:
byte, int, unsigned int, long, char, float, double, uint8_t
An int is 2 bytes long, a long is 4 bytes, a char is 1 byte, a float is 4 bytes, a double is the same as a float, a uint8_t is 1 byte and only holds 0-254.

Inline question: What is the difference between a regular and an unsigned integer?

(unsigned integer only holds positive numbers, a regular int holds positive and negative numbers)

Variables can be in arrays. Char myString[11] holds a 10 character string with a 0 termination

Scope: global vs local

Variables declared outside of functions like setup() will be available to all functions. Variables declared inside a function will only be available inside that one function.

Branching

```
// single branch
if(condition) {
// statements
}

// if this then that
if(condition) {
// condition true statements
} else {
// condition false statements
}

// nested if else ladder
if(condition) {
// condition true
} else if ( condition ) {
//
}

switch (int) {
        sase constant1:
```

```
        // constant1 true statements
        break;
        case constant2:
        break:
        default:
        // if no other case is true
}

// Ternary operator
(condition) ? true statements : false statements ;
```

Inline question: List two ways to test if a counter is above 40, and reset to 0 if it is.

```
if( counter>40) {
        counter = 0;
} else {
        counter++;
}
```

Or

```
(counter>40) ? counter=0 : counter++;
```

Looping

Looping repeats code multiple times

```
// Fixed number of times
for ( init; condition; increment ) {
  // statement(s);
}
Example:
for(i=0; i<10; i++ ) {
        totalLoops++;

}

// Variable number of iterations
while(condition) {
   statement(s);
}
Example:
```

```
while( totalCount < 100 ) {
        totalCount += 10;
}

// Tests condition at the end
do {
  statement(s);
} while( condition );
```

Inline question: Write a loop from 0 to 100 three ways

Grouping Code

To reuse code and to break code into manageable segments, C uses functions and C++ uses objects and methods.

```
// C function

data_type function FunctionName( arguments) {
        Return someValue;
}
Example:
unsigned long function theTimeElapsed( unsigned long offset) {
        return millis() + offset;
}

// To group variables together C offers the Struct

 struct theTime {
   Int minutes;
   Int hours;
   Int seconds;
 };

 struct theTime time_a, time_b;
```

C++

The C++ language includes C as a subset but adds additional capability using objects. An object is an instance of a class. A class contains variable only available to the object, and

variables that are visible outside the object. A class also defines methods which are functions that belong to the object.

Most Arduino libraries are developed to produce objects.

```
// A class definition
class MySerialPort {
// private variables
    int baud_rate, TXpin, RXpin;

// public variables and methods
  public:
    void set_pins (int,int);
    int getRX(void);
};

// Example of a method of the class MySerialPort
void MySerialPort:: set_pins(int rx, int tx) {
          RXpin = rx;
          TXpin = tx;
}
```

In the main program an instance of the MySerialPort is declared which creates an object of the class:

```
MySerialPort myPort;

// A method of the newly created object is called, for example, by:
myPort.set_pins( 9,10);
```

The Arduino IDE has standard classes such as Serial that can be used in the sketch. The IDE automatically creates an instance of Serial for the programmer. With custom libraries the programmer must usually create an instance of the object such as Servo myServo;

Memory Usage

As the Arduino IDE downloads the sketch, it will display the amount of memory for global variables, code and program memory available.

Global variables are convenient but if the global space available runs out then variables can be converted to local variables.

Int variables are the default, but to save space programmers can use “uint8_t” to save 1 byte of memory.

Arduino Extensions to C/C++

(See the Arduino reference or cheat sheet)

Serial,
DigitalWrite
DigitalRead
AnalogRead
AnalogWrite
Time
Math
Servo
Software Serial
Wire

Inline question: How to read milliseconds since the Arduino started?

Debugging

In the terminal window at the bottom of the IDE are any error messages that occur during compilation and uploading. For ease of reading, these messages can be copied to the clipboard and then pasted into a text editor.
If the program compiles and uploads successfully, but doesn’t produce the correct results, the most common way to debug is place “Serial.println()” statements in the code and look at the error messages. The C function sprintf() can be used to combine several variables into one string.

If the internet is available, there is an Arduino simulator which can be used to debug.
https://www.sites.google.com/site/unoardusim/

A search for “online Arduino simulator” yields several websites that offer simulators which can be used for debugging.

Inline question: Where to put a Serial.println() statement?

Consider the code:
i=0;

```
while( i<100) {
        i += 15;
}
```
Where would you place a debugging statement to view every value that the variable “i” takes on?

Exercises

Example 1:

Hardware: No additional hardware required besides an Arduino.
Software: A text editor such as Notepad++ or Pluma desired..

Load the example “Blink”. Save the modified Blink sketch as MyBlink and see the location directory. Then exit the IDE.

Open the MyBlink.ino file with Notepad++ and set variables for the duration number of blinks.

Open the Arduino IDE again, then download and run the modified blink on the Arduino Uno.

```
int timeOn = 500;
int timeOff = 200;
int numBlinks = 1;

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
int i;
for(i=0;i<numBlinks;i++) {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the
voltage level)
  delay(timeOn);                       // milliseconds on
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the
voltage LOW
  delay(timeOff);                       // off time
}
delay(3000);
(numBlinks>9) ? numBlinks=1 : numBlinks++ ;
```

```
}
```

Example 2:

Read a single digit number from serial monitor and blink the onboard LED a number of times.

Software: screen (linux), PuTTy (windows), Terminal (Mac)
Hardware: No additional hardware required besides an Arduino.

```
int ledPin = LED_BUILTIN;

void setup () {
   Serial.begin(9600);
   pinMode(ledPin, OUTPUT);
}
void loop() {

byte oneChar;
int oneNumber;
int i;
unsigned long theTime;

if (Serial.available()) {
   oneChar = Serial.read();
   oneNumber = oneChar - '0'; // converts to integer 0 to N

   if( oneNumber>0 && oneNumber<10) {
       Serial.println(oneNumber);  // echo the input
       for(i=0;i<oneNumber;i++) {
// blink LED
          digitalWrite(ledPin,HIGH);
          delay(300);
          digitalWrite(ledPin,LOW);
          delay(300);
       }
    }
    theTime = millis();
    Serial.print("MS since start: ");
    Serial.println(theTime);
    delay(1000);
   }
}
```

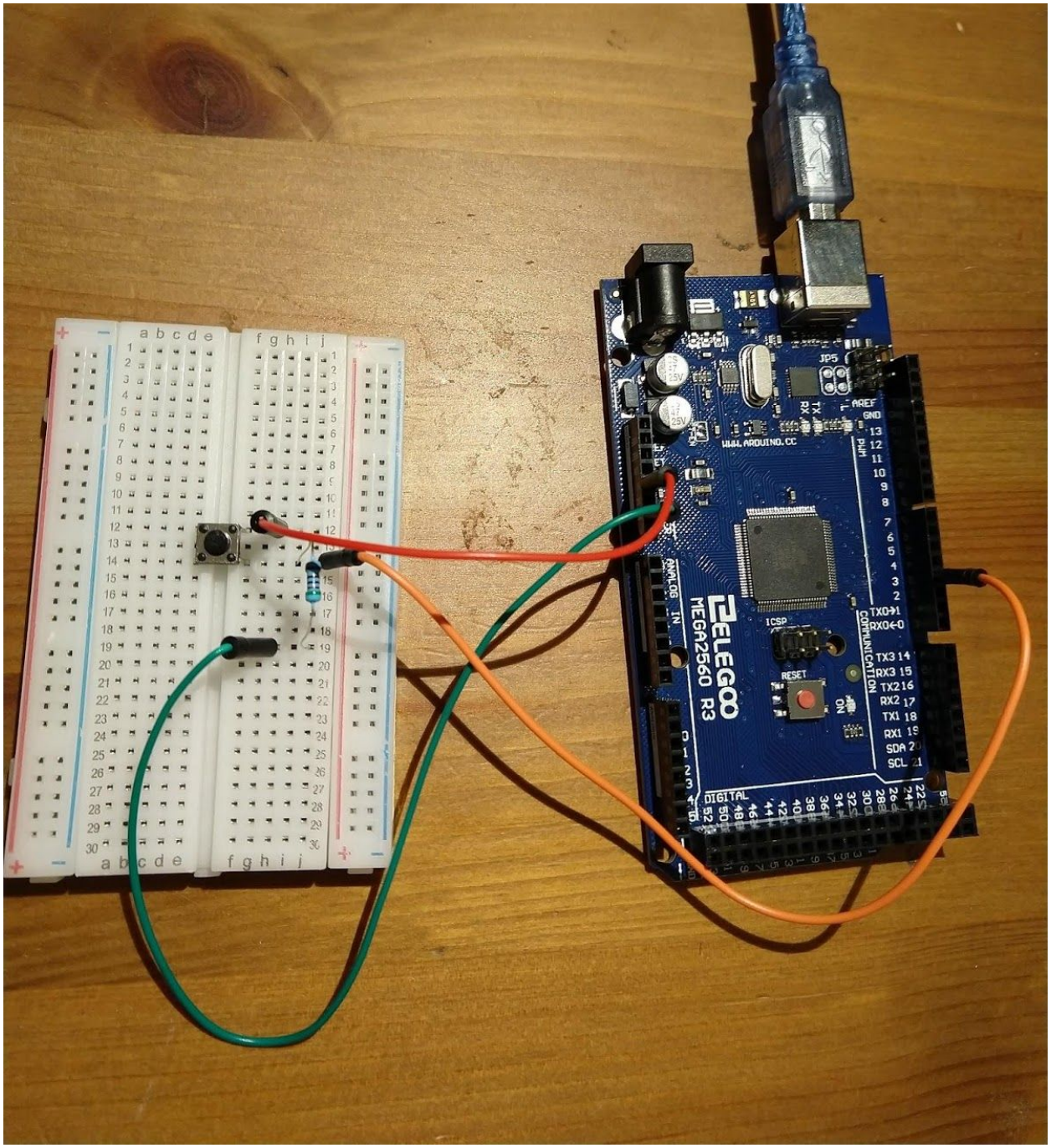
Part 2:
Close the IDE and use a terminal emulation program to communicate with the Arduino.
Software: screen, PuTTY, Terminal

On Linux or rasbian: screen /dev/tty... 9600
On windows: PuTTY.
On MacOS: Terminal

Example 3:

Attach an interrupt to a pin wired to a button. When the button is pushed read the current time in milliseconds and write to serial monitor.

Hardware: breadboard, pull down resistor, 2 wires, and push button.
Software: No additional software needed.




```
// Arduino Programming Workshop Example 3
// Hardware is a button on pin 3 with a pull down resistor
//   to ground. Button goes to +5V when pushed

// pin for the LED. 13 is usually the built-in LED
const uint8_t ledPin = LED_BUILTIN;
```

```
const uint8_t interruptPin = 3;  // For UNO must be 2 or 3
volatile uint8_t toWrite = LOW;

void writeIt();  // function prototype for interrupt function

// Called once on startup
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
// this attaches the interrupt function to a particular IO pin
  attachInterrupt(digitalPinToInterrupt(interruptPin), writeIt, RISING);
  Serial.begin(9600);
}

// Main loop function which is called forever
void loop() {

  unsigned long theTime;

// If an interrupt happened and the toWrite variable changed state
//    write the time to serial monitor

  if (toWrite==HIGH) {

      digitalWrite(ledPin, HIGH);
      theTime = millis();
      Serial.print("Button pressed at ");
      Serial.println(theTime);
      delay(300);
      toWrite=LOW;
      digitalWrite(ledPin, LOW);
  }

  delay(50);
}


// this is the function to call from the interrupt. Must be really short.
void writeIt() {
 // Set flag variable to HIGH
  toWrite = HIGH;

}
```