

Arduino Programming

ME 5411: Mechatronics

February 1, 2017

Lecture Overview

- 1 Overview
- 2 Programming Syntax
 - Data Types
 - I/O & Serial Commands
 - Arduino Program Structure
 - Control Structures
 - Functions
 - Variable Scope
- 3 Compiling & Terminology
- 4 Programming Tips & Tools
 - Notes on Programming
 - Programming Tools
 - Flowcharts
 - Finite State Machines
- 5 Resources

Microcontroller Programming

- Things a microcontroller can do:
 - Remember everything (memory)
 - Fast mathematical and logic calculations
 - Can make decisions based on complex calculations
 - Can perform operations repeatedly
 - All this accomplished by performing *programs*
- **Understand the microcontroller**

What is a Program?

A program is a *sequence of instructions*, and the associated data values, that the computer hardware uses to carry out an *algorithm*

- An *algorithm* is a step-by-step procedure to accomplish something.
- Examples of algorithms:
 - Adding two numbers
 - Finding the largest number in a list
 - Determining the color of an object
 - Localizing a robot

A *programming language* is a formal computer language used to create programs to control the behavior of a machine or to express algorithms.

Types of Programming Languages

- Computers directly execute *machine code*, which gives instructions to perform very specific tasks by directly accessing locations in the computers memory.
- *Assembly language* is a low-level language that closely resembles machine code. This is typically platform-specific.
- *High-level languages*, such as Arduino, C/C++, Java, MATLAB, etc. are easy for humans to work with. These are *compiled* down to machine code so we don't have to worry about it. Generally, these are also portable across platforms.

Comments

Comments let you annotate your code without modifying it. This lets other people (as well as you) understand what your code is doing.

Write comments *as you write your programs!*

```
int num = 3; // Writes a comment that ends at the end of this line
/* This let's you write a comment
   * on multiple lines
  */
```

Data Types

Unlike MATLAB, Arduino (and most other languages) require you to specify the type of variable you are using.

```
// Single values:
```

```
int integer = 3; //stores an integer; cuts off anything after decimal
```

```
float decimal = 3.0; //stores a floating point number
```

```
char letter = 'A'; //stores a single letter. This is stored as a number!
```

```
char letter2 = 65; //this is the same as letter = 'A'
```

```
bool isTrue = true; // true/HIGH/1 or false/LOW/0
```

```
//Arrays:
```

```
int ints[] = {4, 12, 13, -2}; //stores an array of integers
```

```
float floats[3] = {4.0, -3.9, 14.7}; //directly specify number of elements
```

```
char sentence[] = "arduino"; //stores an array of characters, known as a "string"
```

Data Types

- *char*: 1 byte, used to store characters
- *byte*: 8 bit, unsigned number, 0 – 255
- *int*: 16 bit, –32,768 to 32,767
- *unsigned int*: 16 bit, 0 to 65,535
- *long*: 32 bit, –2,147,483,648 to 2,147,483,647
- *unsigned long*: 32 bit, 0 to 4,294,967,295
- *float*: 32 bit, 6-7 decimal places
- *double*: same as float on most Arduinos, 64-bit on a select few
- *bool*: 1 byte, true or false

Data Types & Arithmetic

Be careful combining different data types in arithmetic, they may not behave as you expect them to!

```
int a = 15;  
int b = 2;  
float c = a/b; //truncates the decimal point  
float d = (float)a/b; //cast a to a float to get correct value  
Serial.println(c);  
Serial.println(d);
```



COM3 (Arduino/Genuino Uno)

7.00

7.50

Accessing Array Elements

Arduino is zero-based. Array elements range from 0 to $n - 1$.

```
int ints[5] = {1, 3, 5, 7, 9};  
Serial.println(ints[0]); //returns first element  
Serial.println(ints[4]); //returns last element  
Serial.println(ints[5]); //returns unknown value - out of range
```

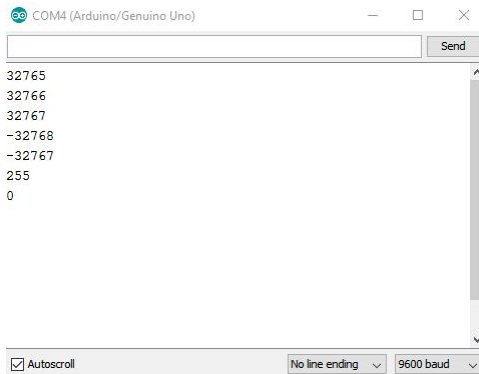
Variable Overflow

If a value is incremented over the maximum value of its data type (or decremented below its minimum), it wraps around to the minimum (or maximum) value.

Saw this a lot on Monday, especially with the loops governed by an if statement.

```
int num = 32765;
Serial.println(num);
num++;
Serial.println(num);
num++;
Serial.println(num);
num++;
Serial.println(num);
num++;
Serial.println(num);

byte b = 255;
Serial.println(b);
b++;
Serial.println(b);
```

A screenshot of the Arduino IDE's serial monitor window, titled 'COM4 (Arduino/Genuino Uno)'. The window shows a list of values printed from the serial port: 32765, 32766, 32767, -32768, -32767, 255, and 0. The values -32768 and -32767 represent the minimum values for a signed integer after overflow. At the bottom of the window, there are settings: 'Autoscroll' is checked, 'No line ending' is selected from a dropdown, and '9600 baud' is selected from another dropdown.

```
COM4 (Arduino/Genuino Uno)
```

```
32765
32766
32767
-32768
-32767
255
0
```

☒ Autoscroll No line ending 9600 baud

Variable Overflow

```
int counter = 0;
void loop() {
  Serial.println(counter);
  if(counter < 50) {
    Serial.println("Hello");
  }
  counter++;
  //will overflow back to 0
}
```

```
int counter = 0;
void loop() {
  Serial.println(counter);
  if(counter < 50) {
    Serial.println("Hello");
    counter++; //stops at 50
  }
}
```

I/O Commands

Arduino uses a number of special commands to access the I/O pins.

```
int ledPin = 13;
int buttonPin = 12;

void setup() {
  //setup specified pin as INPUT or OUTPUT:
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

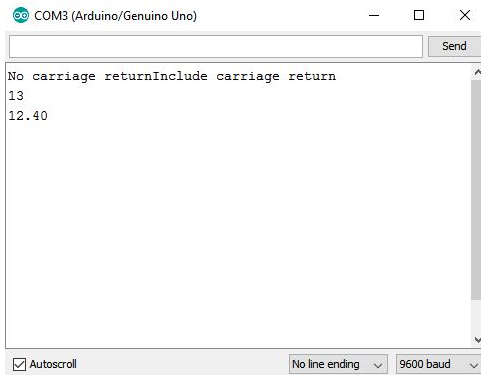
void loop() {
  digitalWrite(ledPin, HIGH); //set LED on
  delay(1000); //wait a second
  digitalWrite(ledPin, LOW); //set LED off
  delay(1000); //wait a second

  digitalWrite(buttonPin); //get value of buttonPin
}
```

Serial Commands

Arduino uses the *Serial* console to log information.

```
Serial.begin(9600); //start at given baud rate
Serial.print("No carriage return");
Serial.println("Include carriage return");
int num = 13;
float val = 12.4;
Serial.println(num);
Serial.println(val);
```

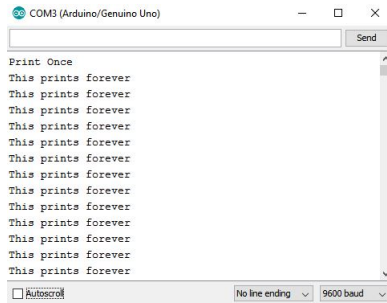


Setup & Loop

Every Arduino program needs to have two functions:

- `setup()` contains things that need to be done once at the start of the program
- `loop()` repeats forever until the program is closed

```
void setup() {  
  // put your setup code here, to run once:  
  Serial.begin(9600);  
  Serial.println("Print Once");  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  Serial.println("This prints forever");  
}
```



Definitions & Includes

At the top of a program, you can include libraries or hard-code constant variables.

```
// Define and Include BEFORE setup()
```

```
//include libraries from Arduino path
```

```
#include <Stepper.h>
```

```
#include <Servo.h>
```

```
//include library from current working directory
```

```
#include "Webserver.h"
```

```
//define some constants
```

```
#define PI 3.14159
```

```
#define TWOPI 2*PI
```

```
void setup() {
```


If Statements

If statements allow for a number of different choices to be executed based on a conditional statement.

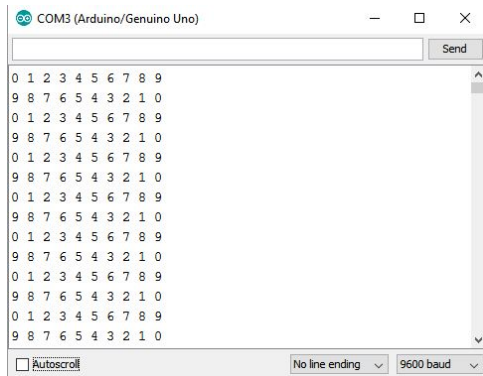
```
if (conditionA) // ==, <, <=, >, >=, !=
{
    something
}
else if (condition B)
{
    something
}
...
else
{
    something
}
```

Once a condition is met (and the associated section of code is executed), the structure ends.

For Loops

A for loop is used to repeat a block of statements. Usually a counter is used to increment and terminate the loop.

```
void loop() {  
  for(int x = 0; x < 10; x++) {  
    Serial.print(x);  
    Serial.print(" ");  
  }  
  Serial.println();  
  for(int x = 9; x >= 0; x--) {  
    Serial.print(x);  
    Serial.print(" ");  
  }  
  Serial.println();  
}
```



```
COM3 (Arduino/Genuino Uno)
```

Send

0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0

☒ Autoscroll

No line ending

9600 baud

While and Do-While Loops

While loops will loop infinitely until the expression inside the parenthesis becomes false.

Do-While loops are the same, with the exception that the condition is tested at the end of the loop.

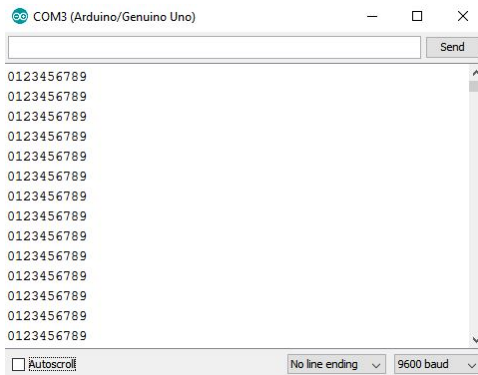
```
void loop() {  
  int x = 0;  
  while (x < 10)  
  {  
    Serial.print(x);  
    x++; //increment x  
  }  
  Serial.println();  
  
  x = 0;  
  do {  
    Serial.print(x);  
    x++;  
  } while (x < 10);  
  Serial.println();  
}
```

While and Do-While Loops

While loops will loop infinitely until the expression inside the parenthesis becomes false.

Do-While loops are the same, with the exception that the condition is tested at the end of the loop.

```
void loop() {  
  int x = 0;  
  while (x < 10)  
  {  
    Serial.print(x);  
    x++; //increment x  
  }  
  Serial.println();  
  
  x = 0;  
  do {  
    Serial.print(x);  
    x++;  
  } while(x < 10);  
  Serial.println();  
}
```



Switch Statements

Switch statements are kind of like if statements. They allow you to compare a variable to a number of cases, with accompanying sections of code to execute:

```
int var;  
switch (var) {  
  case 1: // if var = 1  
  {  
    //do something  
    break;  
  }  
  case 2: // if var = 2  
  {  
    //do something  
    break;  
  }  
  ...  
  default: //if nothing else matches,  
           //do this  
  {  
    //do something  
    break;  
  }  
}
```

Note: break keyword is used to prevent “falling-through”

Functions

If you would like to perform a task more than once, you can write a function to do it.

```
void loop() {  
    int num = 13;  
    int signNum = sign(num); //call function  
}  
  
int sign(int num) {  
    if (num > 0) {return 1;}  
    else if (num < 0) {return -1;}  
    else {return 0;}  
}
```

Functions

- A function needs to specify its inputs and its outputs.
 - If there are no inputs, just leave the paranthesis empty ().
 - There is a special output type called “void” for when there are no outputs.
- The “return” command is used to give the output. Once return is called, the function ends.
- Functions need to be declared before they can be used. (Can be “prototyped”)
- `setup()` and `loop()` are both functions, although they are special (and necessary) in Arduino

Prototyping Functions

```
int sign(int num); //prototype function here

void loop() {
    int num = 13;
    int signNum = sign(num); //call function
}

int sign(int num) { //specify function here
    if (num > 0) {return 1;}
    else if (num < 0) {return -1;}
    else {return 0;}
}
```


Variable Scope

Variables have a property called “scope”, which basically means where the variable is accessible.

- A variable declared before *setup()* is accessible anywhere.
- A variable declared within a function (*setup()*, *loop()*) is only accessible within that function.
- A variable is *never* accessible before it is declared.

Variable Scope

```
int pin = 13; //this is accessible anywhere

void setup() {
  Serial.begin(9600);
  Serial.println("In setup function");
  Serial.println(pin);

  int pin2 = 7; //this can only be seen in setup()
  Serial.println(pin2);
}

void loop() {
  Serial.println("In loop function");
  Serial.println(pin);
  Serial.println(pin2); //what is this?
}
```

'pin2' was not declared in this scope

Copy error messages

C:\Users\Michael\Documents\Arduino\sketch_jan30a\sketch_jan30a.ino: In function 'void loop()':

sketch_jan30a:15: error: 'pin2' was not declared in this scope

```
    Serial.println(pin2); //what is this?
```

exit status 1

'pin2' was not declared in this scope

Compiling Arduino Code: What's Happening?

- Typically, when we work with Arduino code, we do so in the IDE.
- When we press the arrow button, the code is compiled into binary (machine code).
- The compiled code is then stored into ROM (in Arduino's case, it is stored in the flash memory)

Programming Terminology

- **Flash burner:** utility that is used to burn the generated code into the memory of the microcontroller
- **Bootloader:** a preprogrammed software that does initializations and configures basic I/O peripherals before the user-code can start executing.
- Note: Arduino IDE provides all these in a combined environment. It already has a bootloader burned into the flash.

Notes on Programming

- Programming is *expensive* - typically takes 1.5 hours of programming per instruction when starting from scratch
 - Most (industrial) programs have several thousand instructions
- “Bugs” (errors in a program) are hard to avoid
 - Programs with bugs are typically useless
 - Bugs can be really, really hard to track down and fix
 - This can lead to frustration
 - “You either like programming or you don't ” (I do).
 - Frustration can be amplified if you just resort to trial and error.

In order to minimize frustration, there are a few tips and methods (tools).

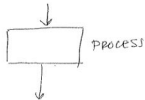
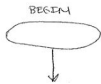
Reasonable Programming Goals

- Try to keep programs as short as necessary
 - Arduino has limited program length - 500 lines
 - Not very practical
 - May be too expensive or inhibit progress
- Make programs that are easily understandable (use comments!)
- Make programs that are easily modifiable
- Finish programs on schedule - especially for a class!

Flowcharts

Flowcharts are a way to visualize the process of your program without having to write any code.

SYMBOLS



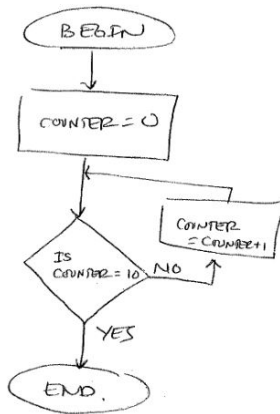
- Start/stop with only one begin/end
- Process: 1 arrow in, 1 arrow out
- Decision: 1 arrow in, 2 arrows out
- If 2 arrows meet, they do so before end

Flowcharts

SIMPLE EXAMPLE

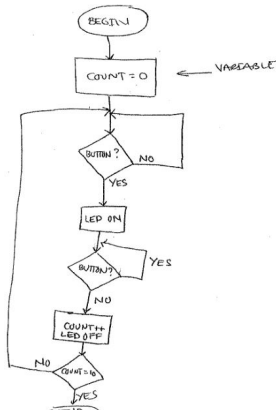
START w/ COUNT = 0

COUNT TO 10 AND END



Flowcharts

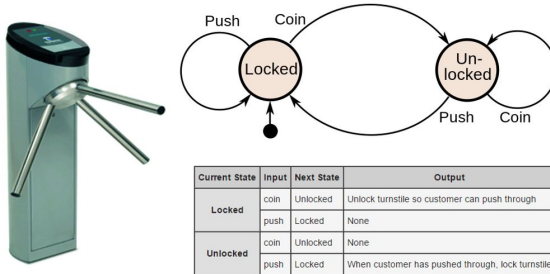
When the user presses (and holds) a button, an LED lights up, otherwise it is off. After it has been lit 10 times, the program ends.



Finite State Machines

A finite state machine is another way to represent logic for a system. It consists of states and transitions.

They system can only be in one state at a time and will change states if an event described in a transition occurs.



Arduino Resources

- Dr. Clayton & Mike Benson
- Experienced classmates
- Arduino Examples (*File > Examples > ...*)
- The internet
 - <https://www.arduino.cc/en/Reference/HomePage>
 - Google, YouTube
 - Product-specific webpages for sensors, etc.

Buttons

