ROBOTICS CRASH COURSE: ARDUINO ESSENTIALS

1. Code to use for your Robot

Install Libraries

```
#include <rcc.h>     //robotics crash course library
#include <Servo.h>   //servo motor library
#include <Wire.h>    //set up I2C for imu sensor library
```

Define Global Constants

```
#define ANGLE 30      //define a global constant
#define POT_PIN A0    //can define pins too (can define anything, really)
```

Declare Global Variables

```
int buttonPin = A1;         //define variables syntax: type name = value;
unsigned long duration_us; //variable for ultrasonic sensor
```

Hardware Setup *(need these lines before void setup())*

```
Servo actuator;       //servo motor
HC_SR04 ultrasonic;   //ultrasonic sensor
Odom odom;            //encoder IR sensor
MPU6050 imu;          //accelerometer & gyroscope sensor
```

Arduino's Setup Function (only runs once)

```
void setup() {

  Setup Serial Monitor
  Serial.begin(9600); //9600 is baud rate (communication rate)

  Pin Modes for Extra Hardware

   pinMode(LED_BUILTIN, OUTPUT);     //setup arduino's hardware pins
   pinMode(A0, INPUT);          //works if you define POT_PIN earlier
   pinMode(A1, INPUT_PULLUP); //button needs pullup

  Peripheral.begin() (Initialize Hardware)
   motorSetup();                   //initialize motor
   actuator.attach(RCC_SERVO_PIN);  //initialize servo motor
   ultrasonic.begin();             //initialize ultrasonic sensor
   odom.begin();                   //initialize encoder hardware
   Wire.begin();                   //setup I2C
   imu.begin();                    //start imu readings
   imu.calibrate();                //keep robot still to calibrate
}
/*end of void setup()*/
```

Arduino's Loop (runs over and over)

```
void loop() {

  Print to Serial Monitor for debugging
  Serial.println("print things with this");

  DC Motor Control
  //(takes values from -255 to 255);
  //inputs are (left, right) motors
  rawMotorCtrl(200, 200);

  Servo Motor
  //to control servo's angle (ranges from 0-180)
  actuator.write(90);

  Ultrasonic Sensor
  //step 1: send out a pulse and count time it takes to return
  duration_us = ultrasonic.pulse();


  //step 2: convert time to distance in cm
  distance = duration2centimeters(duration_us);

  Encoders
  //step 1: get initial count
  unsigned long count_start = odom.getRightCount();

  //step 2: setup current count variable
  unsigned long count_current;

  //step 3: subtract values to get amount of tickmarks since started
  //(make sure you setup your desired count variable too)


  unsigned long desired_count = 10;
   while((count_current = odom.getRightCount() - count_start) <= desired_count)
       {/*do something*/;}


  //note: to use left encoder use odom.getLeftCount();

  Inertial Measurement Unit
  //step 1: read sensor
  imu.update(); //run this to get a new reading

  //step 2: get value you want (acceleration or angular velocity)
   float accel_x = imu.getAccelX();
   float accel_y = imu.getAccelY();
   float ang_vel_x = imu.getAngVelX();
   float ang_vel_y = imu.getAngVelY();
   float ang_vel_z = imu.getAngVelZ();


 } /*end of void loop*/
```

2. LOGICAL STATEMENTS IN C++

If (binary response to condition, think "yes" or "no")

```
if(/*condition is true*/){/*do something*/;}
else if(/*second condition is true*/){ /*do something different*/;}
else{/*do something else*/; }
```

example:

```
if (state == 1){/*do something*/}
```

For Loop (do something for a desired amount of cycles)

```
for (/*counter*/; /*counter limit*/; /*increment*/){}
```

example:

```
for (int i = 0; i < 6; i++) {
   /*do something 6 times*/
 }
```

While Loop (do something while a condition is true)

```
while(condition){/*do something*/;}
```

examples:

```
while(1){/*use this if you want something to always happen*/;}
```

```
while(distance <= 10){/*go forward*/;}
```

More Logic Syntax:

```
Equals "=="   ("=" sets a variable, need "==" to check equality)
AND    "&&"
OR     "||"
NOT    "!"
```

3. CREATING A FUNCTION IN C++

```
//step 1: define function outside void setup() and void loop()

syntax format:
/*variable type returned*/ functionName(/*input(s)*/){
      /*code function will run*/;
   }

//step 2: call function within void setup() or void loop()

example sketch:
```

```
void setup(){
       Serial.begin(9600);
}
void loop(){
       int i = 2;
       int j = 3;
       int k;
       k = myMultiplyFunction(i,j);
       Serial.println(k);
}
int myMultiplyFunction(int x, int y){
      int result;
      result = x * y;
      return result;
} //serial monitor will print "6"
```

another example (pseudo~code):

```
void runRightMotor(int tickmarks){
   /*setup odom variables*/
   while((/*odom count*/ <= tickmarks){/*run right motor*/}
  }

void loop() {
  byte tickmarks;
  // read potentiometer, map to 10-50 ticks
  tickmarks = map(analogRead(POT_PIN), 0, 1023, 10, 50);

  //call function, input is from potentiometer
  runRightMotor(tickmarks);

  //stop motor after the run function
  while(1){rawMotorCtrl(0,0);}
}
```