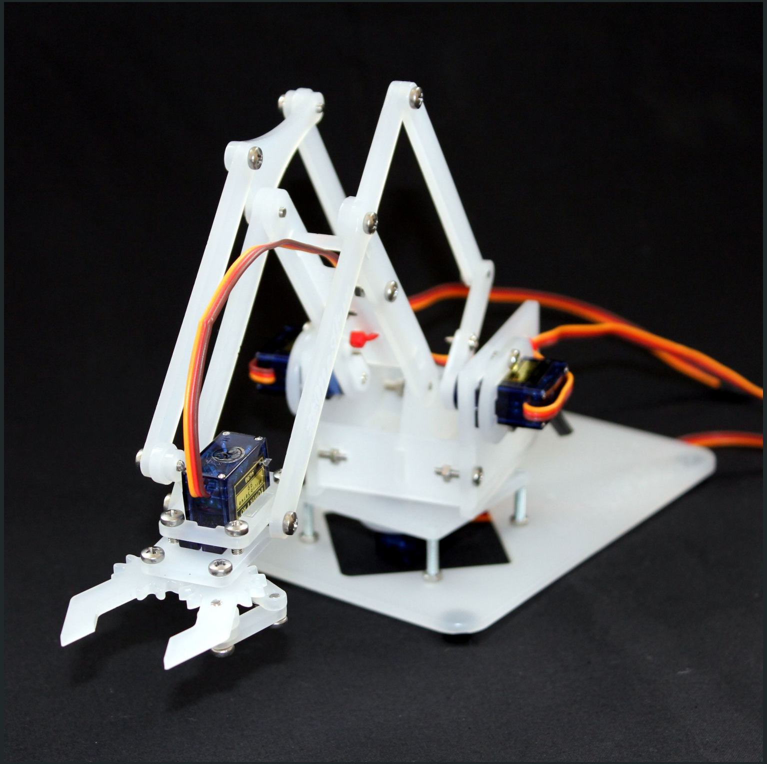
INF1805 - Sistemas Reativos 2017.1

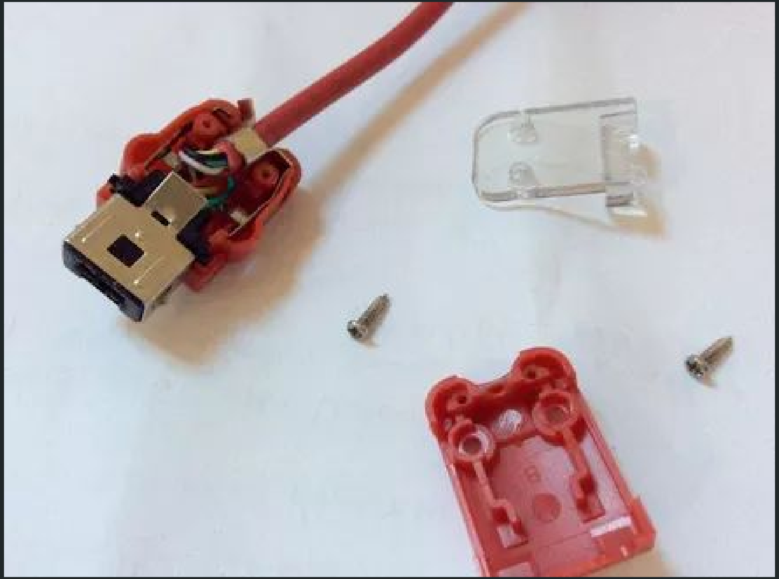
Mini-projeto de Arduino: Controle de Braço Robótico com Nunchuk

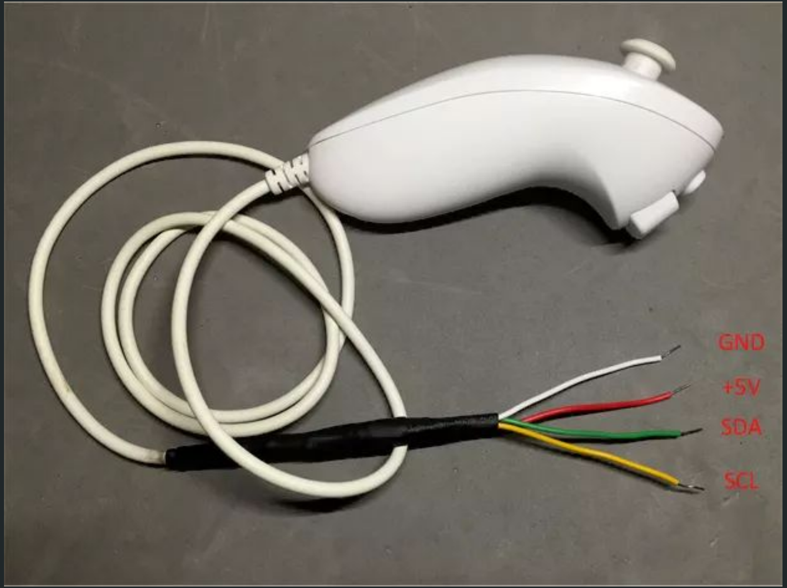
Samuel Bastos

Objetivo



Nunchuk

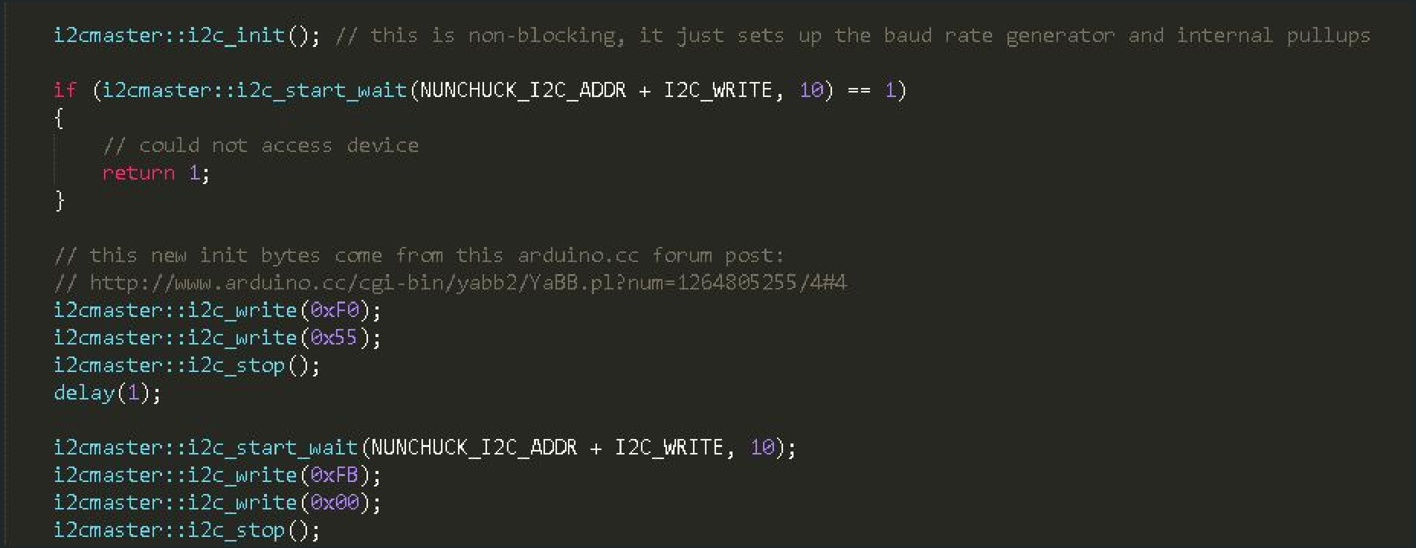




I²C

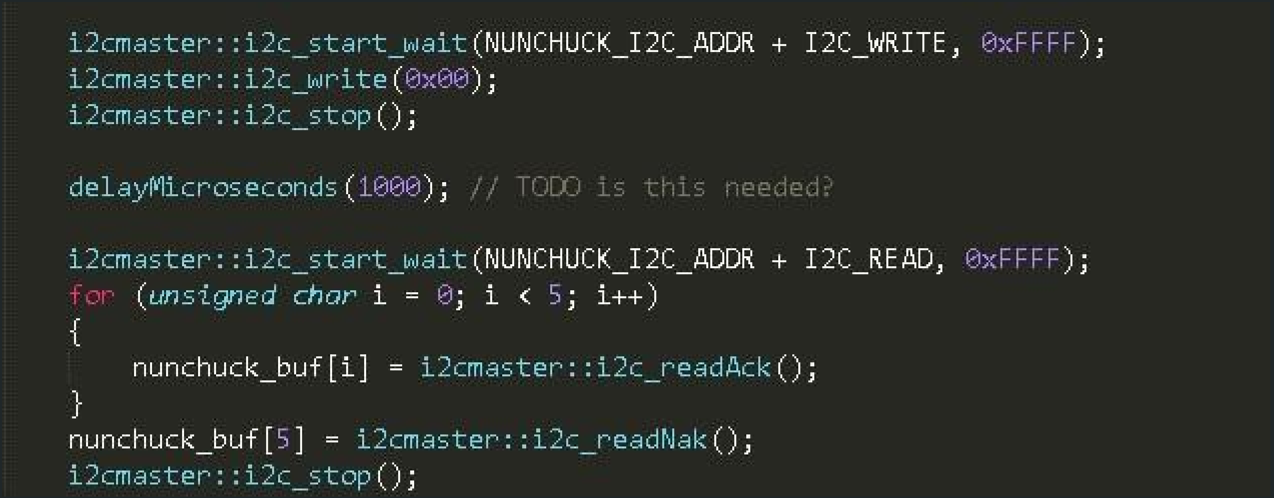
Master Node: Arduino

Slave Node: Nunchuk



```
i2cmaster::i2c_init(); // this is non-blocking, it just sets up the baud rate generator and internal pullups

if (i2cmaster::i2c_start_wait(NUNCHUCK_I2C_ADDR + I2C_WRITE, 10) == 1)
{
    // could not access device
    return 1;
}

// this new init bytes come from this arduino.cc forum post:
// http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1264805255/4#4
i2cmaster::i2c_write(0xF0);
i2cmaster::i2c_write(0x55);
i2cmaster::i2c_stop();
delay(1);

i2cmaster::i2c_start_wait(NUNCHUCK_I2C_ADDR + I2C_WRITE, 10);
i2cmaster::i2c_write(0xFB);
i2cmaster::i2c_write(0x00);
i2cmaster::i2c_stop();
```

I^{2}C



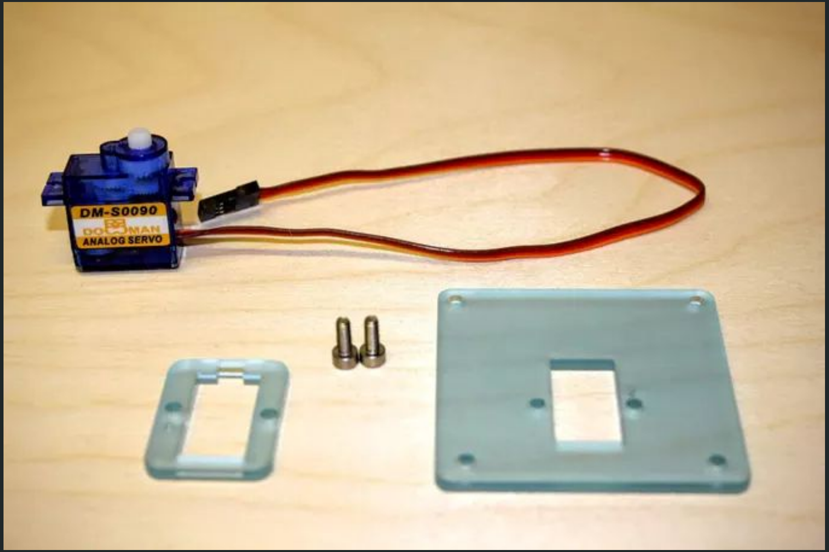
```
i2cmaster::i2c_start_wait(NUNCHUCK_I2C_ADDR + I2C_WRITE, 0xFFFF);
i2cmaster::i2c_write(0x00);
i2cmaster::i2c_stop();

delayMicroseconds(1000); // TODO is this needed?

i2cmaster::i2c_start_wait(NUNCHUCK_I2C_ADDR + I2C_READ, 0xFFFF);
for (unsigned char i = 0; i < 5; i++)
{
    nunchuck_buf[i] = i2cmaster::i2c_readAck();
}
nunchuck_buf[5] = i2cmaster::i2c_readNak();
i2cmaster::i2c_stop();
```

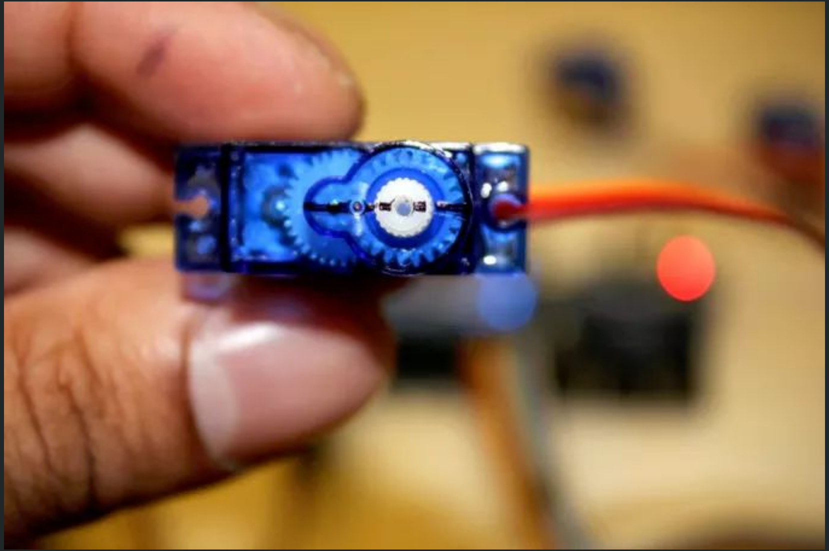
Servos

“Servo control is achieved by sending a servo a PWM (pulse width modulation) signal, a series of repeating pulses of variable width where either the width of the pulse (most common modern hobby servos) or the duty cycle of a pulse train (less common today) determines the position to be achieved by the servo. The PWM signal might come from a radio control receiver to the servo or from common micro-controllers such as the **Arduino**.” - Wikipedia

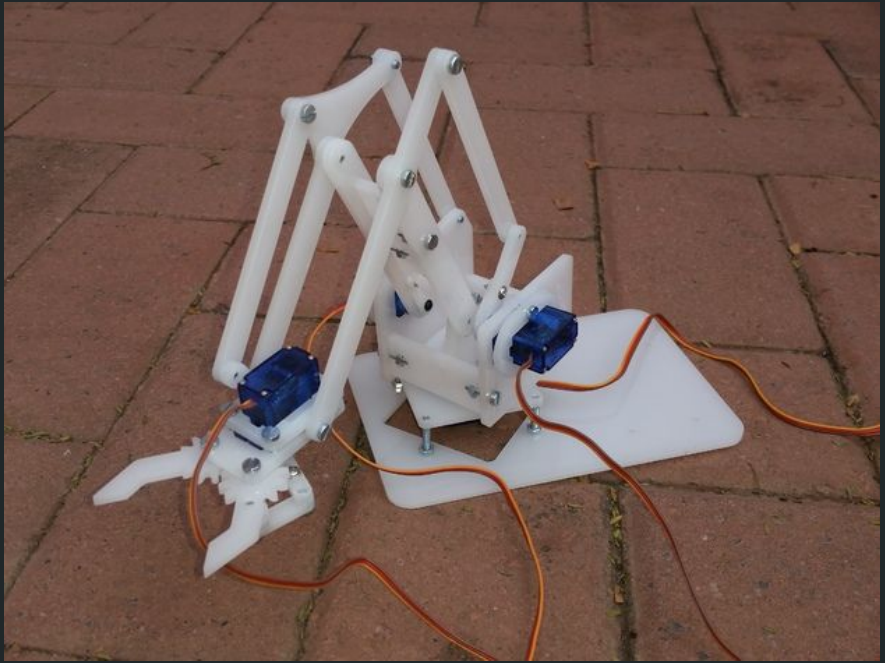
“Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.” - Servo library reference page from Arduino website

MeArm





MeArm

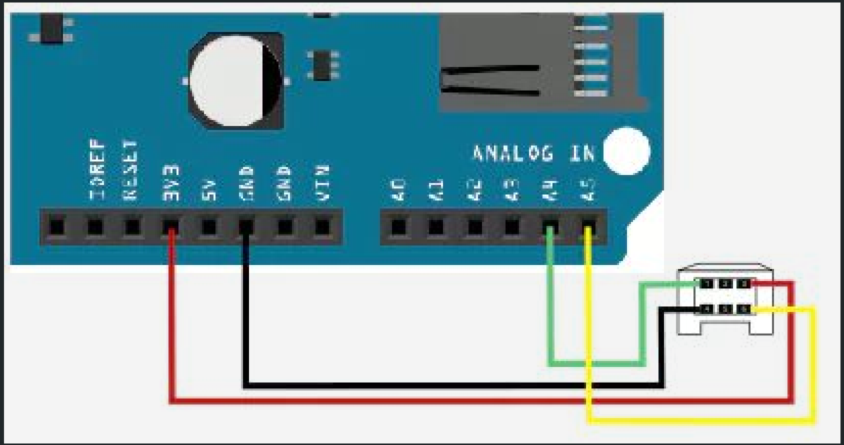


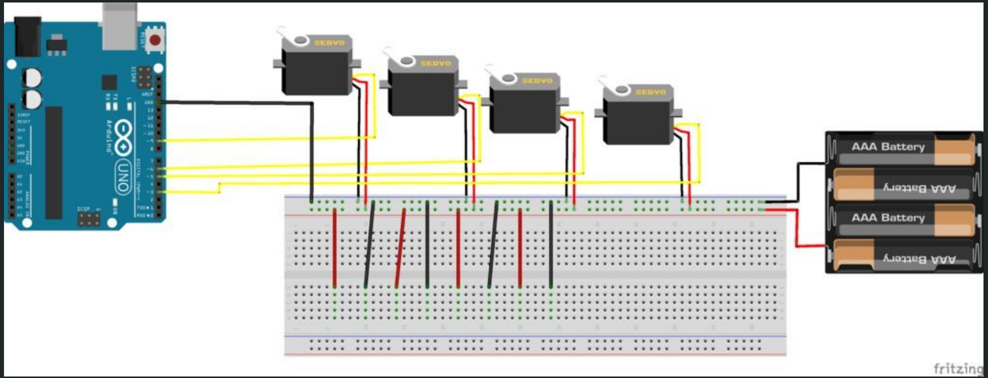
Inverse Kinematics Control Library

The meArm has four mini servos - one for the gripper, and one each to rotate the base, shoulder joint and elbow joint. But it's not terribly convenient to be specifying things in terms of servo angles when you're much more interested in where you would like to place the gripper, in normal Cartesian (x, y, z) coordinates.

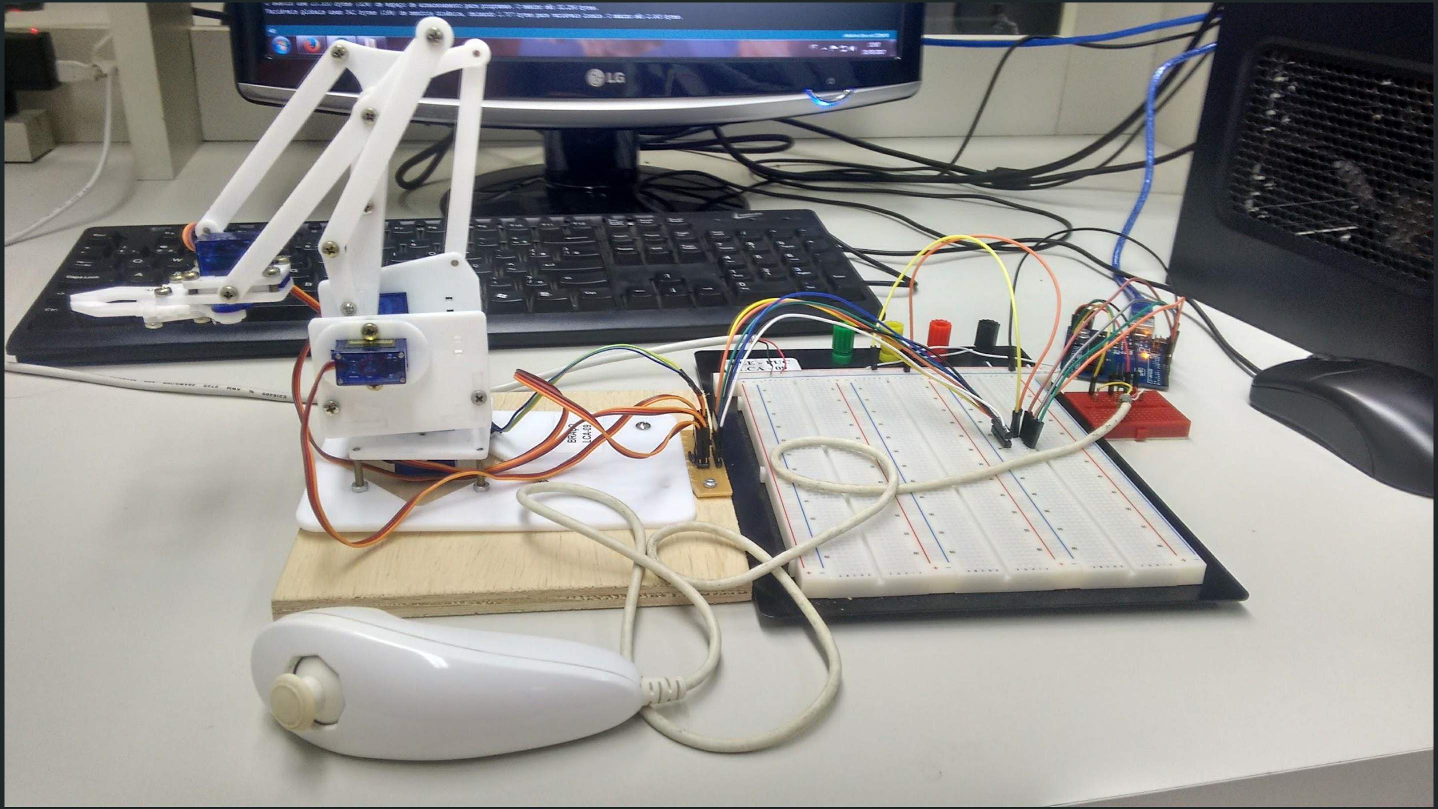
This library solves the angles required to send to the servos in order to meet a given position, allowing for much simpler coding.

Circuito

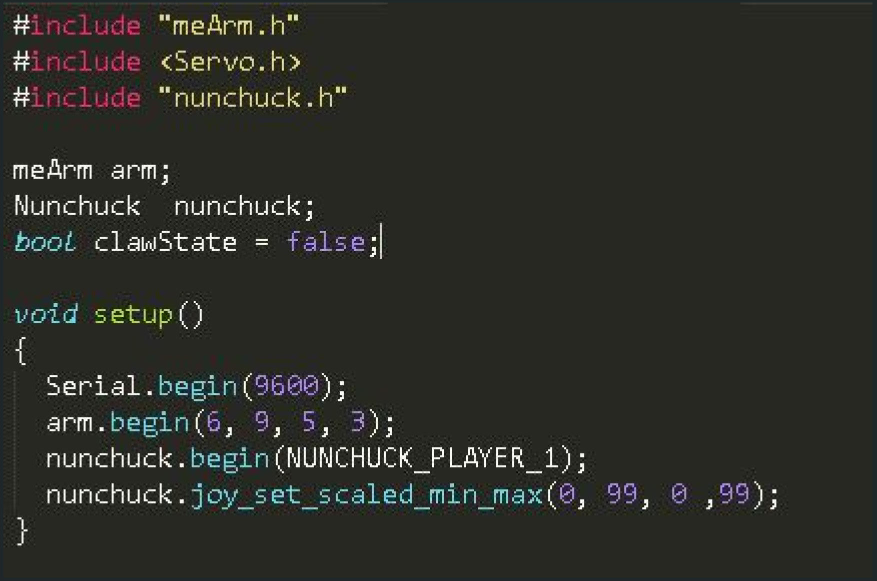

IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5


SERVO
SERVO
SERVO
SERVO
AAA Battery
AAA Battery
AAA Battery
AAA Battery
fritzing

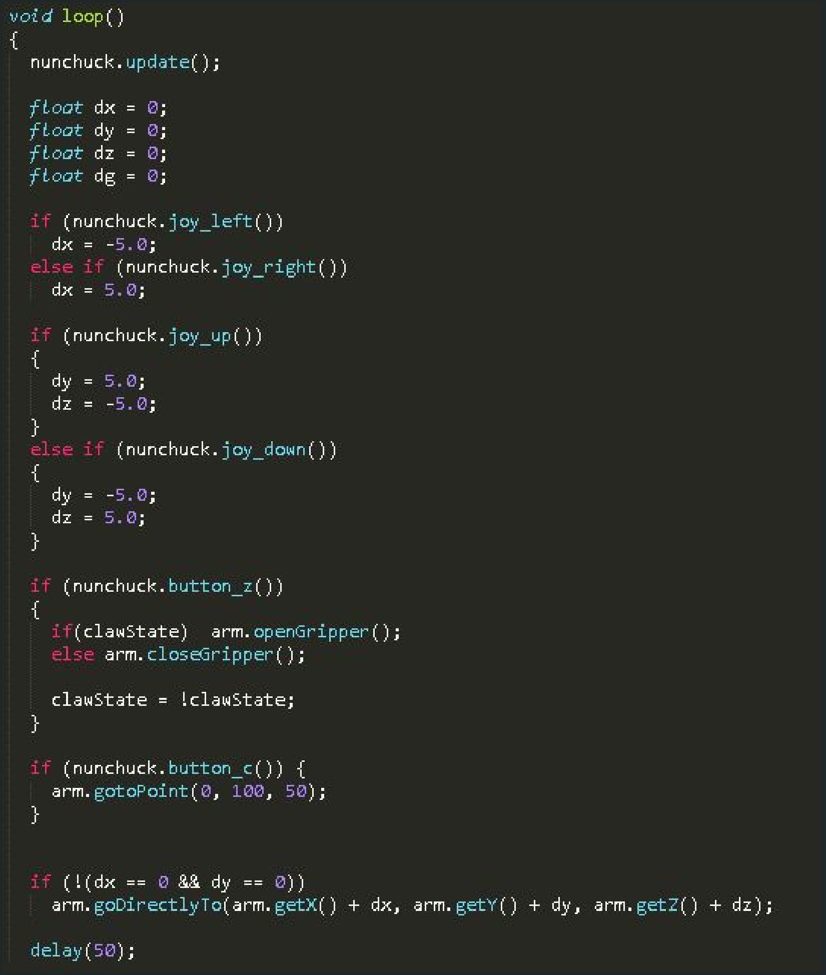
Circuito



Main



```
#include "meArm.h"
#include <Servo.h>
#include "nunchuck.h"

meArm arm;
Nunchuck  nunchuck;
bool clawState = false;

void setup()
{
  Serial.begin(9600);
  arm.begin(6, 9, 5, 3);
  nunchuck.begin(NUNCHUCK_PLAYER_1);
  nunchuck.joy_set_scaled_min_max(0, 99, 0 ,99);
}
```



```
void loop()
{
  nunchuck.update();

  float dx = 0;
  float dy = 0;
  float dz = 0;
  float dg = 0;

  if (nunchuck.joy_left())
    dx = -5.0;
  else if (nunchuck.joy_right())
    dx = 5.0;

  if (nunchuck.joy_up())
  {
    dy = 5.0;
    dz = -5.0;
  }
  else if (nunchuck.joy_down())
  {
    dy = -5.0;
    dz = 5.0;
  }

  if (nunchuck.button_z())
  {
    if(clawState)  arm.openGripper();
    else arm.closeGripper();

    clawState = !clawState;
  }

  if (nunchuck.button_c()) {
    arm.gotoPoint(0, 100, 50);
  }


  if (!(dx == 0 && dy == 0))
    arm.goDirectlyTo(arm.getX() + dx, arm.getY() + dy, arm.getZ() + dz);

  delay(50);
```

Funcionamento do Circuito

Funcionamento do Circuito