BASIC ARDUINO PROGRAMMING

Mads Høbye, Datalogi, Fablab, illutron....
www.hobye.dk

Arduino / Debugging

The arduino interface

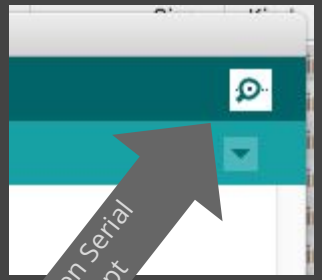
Basic arduino debugging:

Baud rate

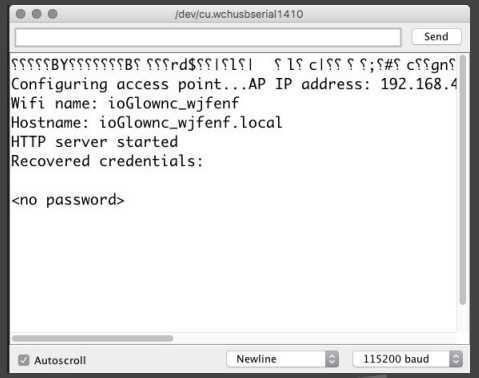
```
float value = 20;

void setup()
{
        Serial.begin(115200);
}

void loop()
{
        Serial.println("The value is: ");
        Serial.println(value);
        delay(100);
}
```

Init serial

Print line



Open Serial prompt



Baud rate

Scopes and methods

Basic arduino template:

setup method

Scope start

Scope end

```
void setup()
{
    // RUNS ONCE IN THE BEGINNING
}
```

loop method

Scope start

Scope end

```
void loop()
{
    // LOOPS ALL THE TIME
}
```

Method without a return:

Does not return anything

Name of method

Two int params

Scope start

```
void myMethod(int param1, int param2)
{
    // code here
    int sum = param1 + param2;
}
```

Call the method: myMethod(2,2);

Method with a return:

returns int

Name of method

Two int params

Scope start

```
int myMethod(int param1, int param2)
{
    // code here
    return param1 + param2;
}
```

Required return call

Call the method: myMethod(2,2);

Nested scopes:

global variable outside scope

Scope start

Local variable inside scope

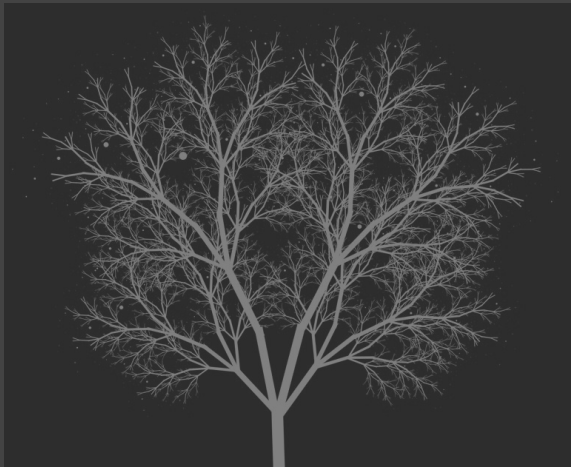
Nested scope start

Resets all the time

```
int globalVariable = 0;

void loop()
{
    int localVariable = 22;
    if(globalVariable == 10)
    {
        localVariable = 100;
    }
}
```

A recursive method:

```
void recursiveMethod(int i)
    {
        If (i > 1)
        {
            recursiveMethod(i-1);
        }
    }
```

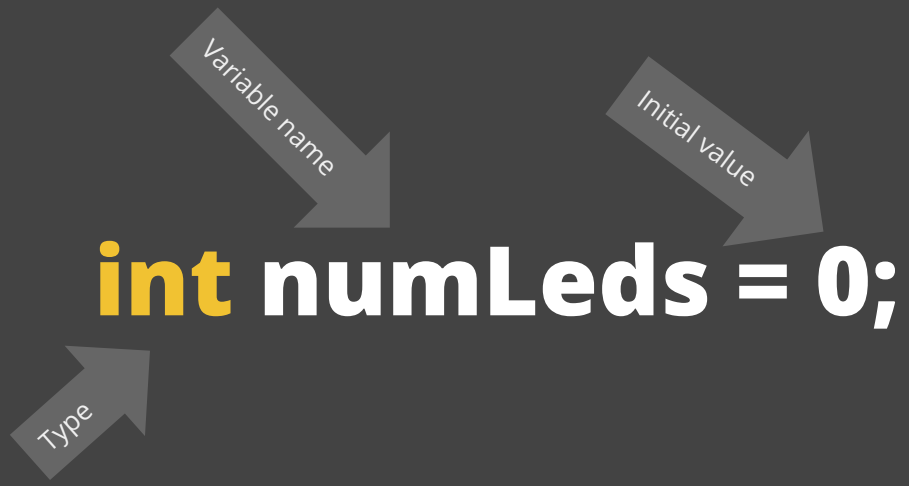


Variables and datatypes

Data types (kinds of variables):

double	3.434345	large decimal number
boolean	true / false	
float	3.555444	decimal number
char	‘a’	A character
int	12123	integer is a whole number
long	123123	long is a large whole number
string	“sdfsdf”	Not really a variable

Defining a variable:

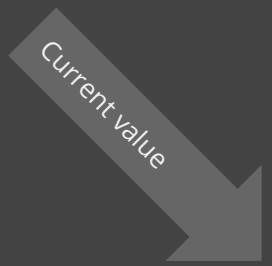


Types: float (10.2), int (10), boolean(true/false), char(0-255)

Changing a variable:

numLeds = 100;

Incrementing a variable:



numLeds = numLeds + 1;

(or)

numLeds++;

Logic and conditions

Conditional statement:

Condition

Scope start

Scope end

```
if(numLeds > 10)
{
    // CODE HERE
}
```

Other conditions: <, >, ==, !=, >=, <=

Else condition:

```
if(numLeds > 10)
{
    // CODE HERE
}
else
{

}
```

Executed if condition is not met

Else if condition:

Executed if the first condition is not met and the new condition is met

```
if(numLeds > 10)
{
    // CODE HERE
}
else if (numLeds == 11)
{

}
else {}
```

Conditional range:


```
if(numLeds > 10 && numLeds < 100)
{

}
```

Other conditions: ||. “||” means OR

For loop (runs for ten times. “i” tarts with zero and ends at 9):

Type

Start number

Repeat until i = 10

Increment one for each loop

```
for(int i = 0; i < 10; i ++)
{
    leds[i] = CRGB(255,255,255);
}
```

using the index variable

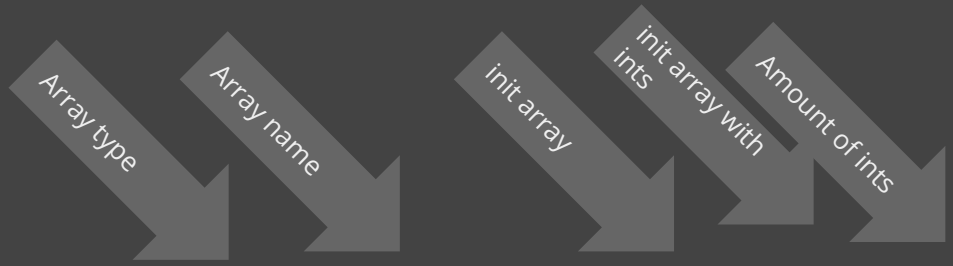
While loop (runs as long as “a” is less than 10):

Condition

```
int a = 0;
while (a < 10)
{
        a =a++;
}
```

Arrays

Creating an array




```
int[] ints= new int[10];
```

Iterating an array:

```
int[] ints= new int[10];

for(int i = 0 ; i < 10; i= i +1) {
    ints[i] = 0;
}
```

Iterate over ints

Set all int to 0

Timing

Make an interval timer:

```
unsigned long timer = 0;
void setup()
{

}

void loop()
{
    if(millis()-timer > 2000) // do something every two seconds)
    {
        timer = millis();
        //do something here
    }
}
```

Timing an event:

At some point you need to time an event. A button has been pressed and you want to do something e.g. two seconds later. To do so you need to register the time the button was pressed and look for when e.g. two seconds has passed from then:

```
long timeWhenPressed = 0;
boolean oldMouseState = 0;
boolean timerActive = false;

void setup(){
        pinMode(1, INPUT);
}

void loop(){

            if(digitalRead(1) == HIGH)    {
                timeWhenPressed = millis();
                timerActive = true;
            }

            if(millis()-timeWhenPressed > 2000 && timerActive == true)     {
                // do something two seconds later
                timerActive = false;
            }
}
```

Events and states

STATE (objective state):
The door is open (or closed).

EVENT (change of state):
The door changed from open to closed.

State: Button <u>is</u> pressed

```
void setup()
{
    pinMode(2, INPUT_PULLUP);
}

void loop()
{
    if(!digitalRead(2) == true)
    {
        // button is pressed.
    }
    else
    {
        // button is not pressed
    }
}
```

Event: Button has been pressed

If you want to do something when the button was pressed => Going from a open state to a closed state. This requires us to "remember" the old state and compare it to detect the change:

```
boolean btnPressedOld = false;
void setup()
{
    pinMode(3, INPUT_PULLUP);
}

void loop()
{
    boolean btnPressed = !digitalRead(3);
    if(btnPressed == true && btnPressedOld == false)
    {
        // button has been pressed.
    }
    btnPressedOld = btnPressed;
}
```

State machine

As the complexity of your code grows you will need to have multiple states where different lines of code should happen:

```
int currentState = 0;


void loop(){
    if(currentState == 0){
        if([condition to change state})
        {
            currentState == 1;
        }
    }
    else if(currentState==1)
    {
    }

}
```

Math and data manipulation

Changing the range of a signal:

Let us say that we have a distance sensor returning a value between 0 and 1024 and we want to move a servo that has the range of 0 to 180. Put simply we can just divide by the difference in the range, but both Arduino and Processing have code to solve this problem.

```
float inputValue = analogRead(0);
float output = map(inputValue,0,1024,0,180);
float output_constrained = constrain(output,0,180);
```

Running average

If your signal is noisy or you want to have a more soft reaction pattern, then you can use a running average.

```
float smoothValue = 0;

void setup()
{}

void draw()
{
    float inputValue = analogRead(0);
    smoothValue = smoothValue * 0.9f + 0.1f * inputValue;
}
```

Input

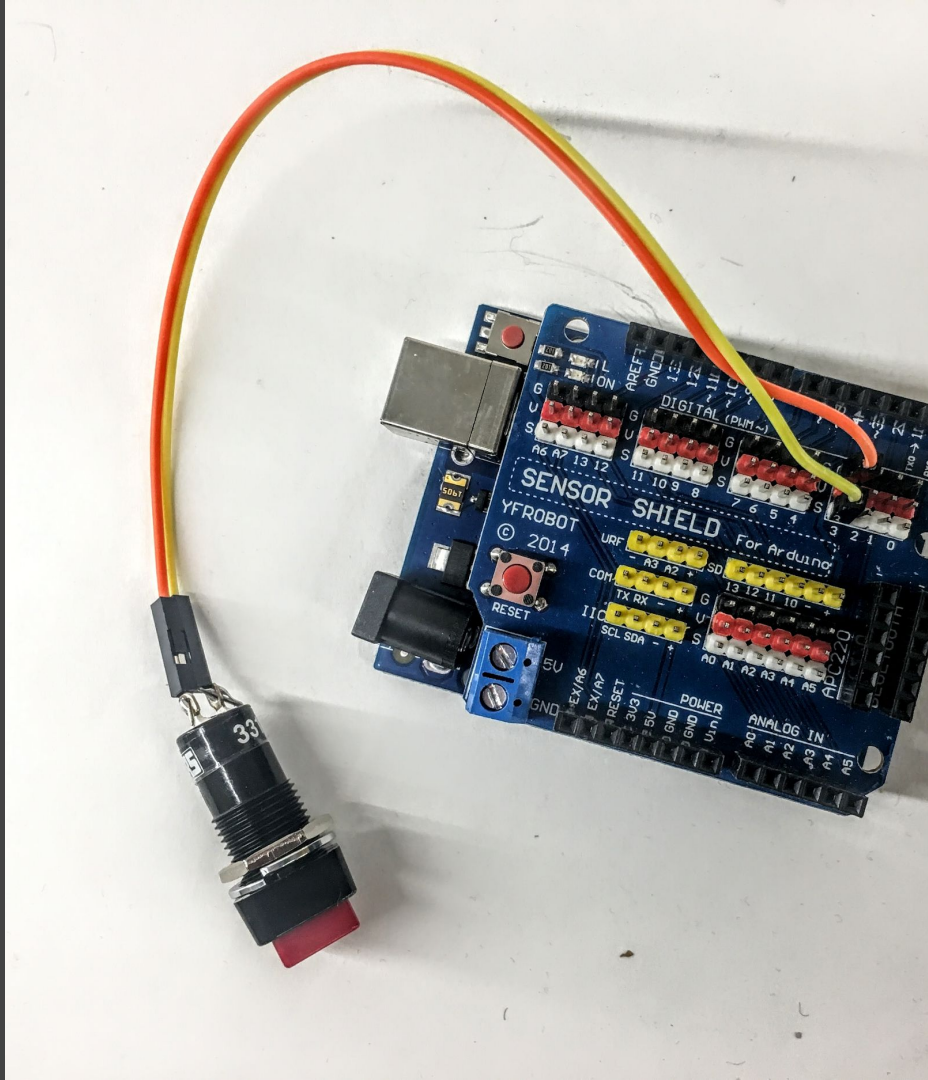
Button pressed

set the pin to input

```
void setup()
{
    pinMode(3, INPUT_PULLUP);
    Serial.begin(9600);

}
```

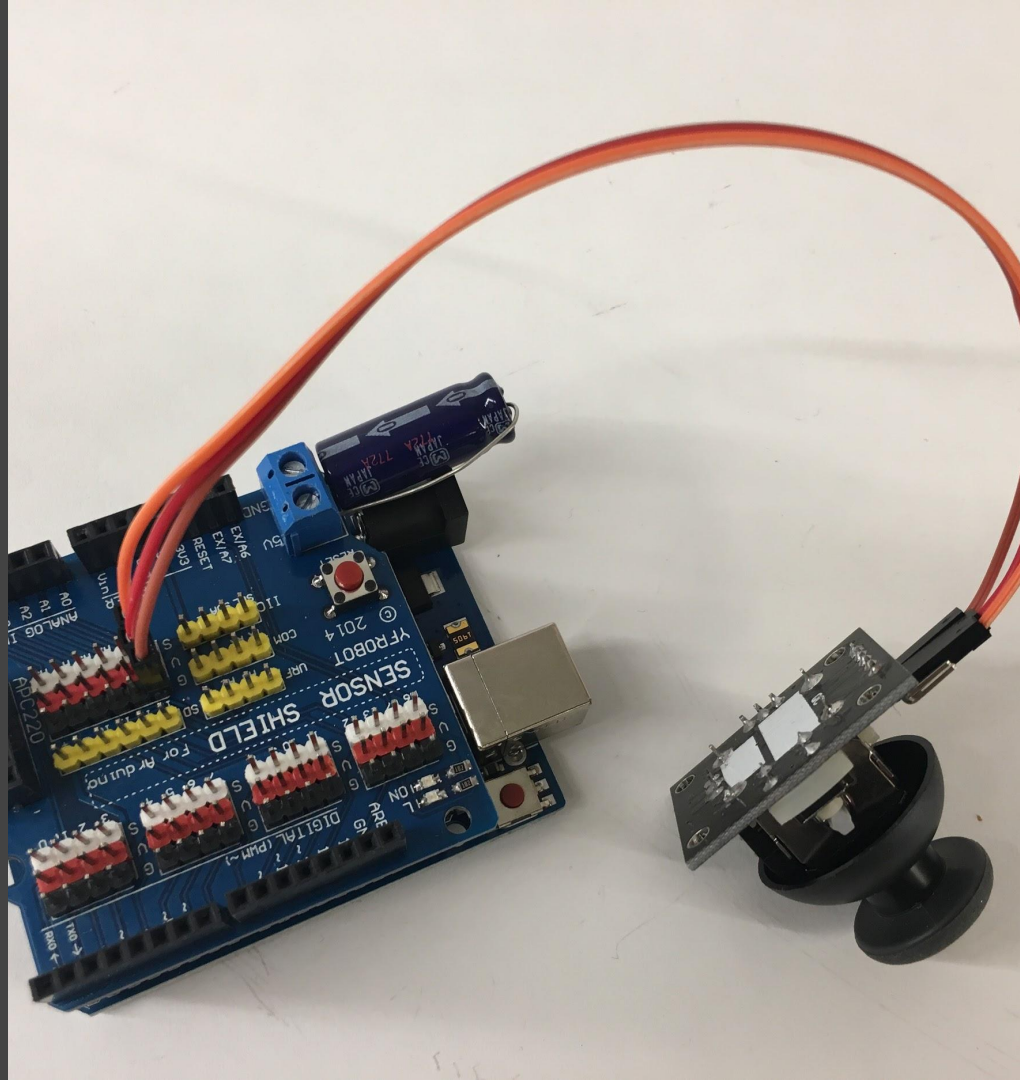
reads the state of the pin

```
void loop()
{
    boolean btnPressed = !digitalRead(3);
    if(btnPressed == true)
    {
        Serial.println("button pressed.");
    }

}
```



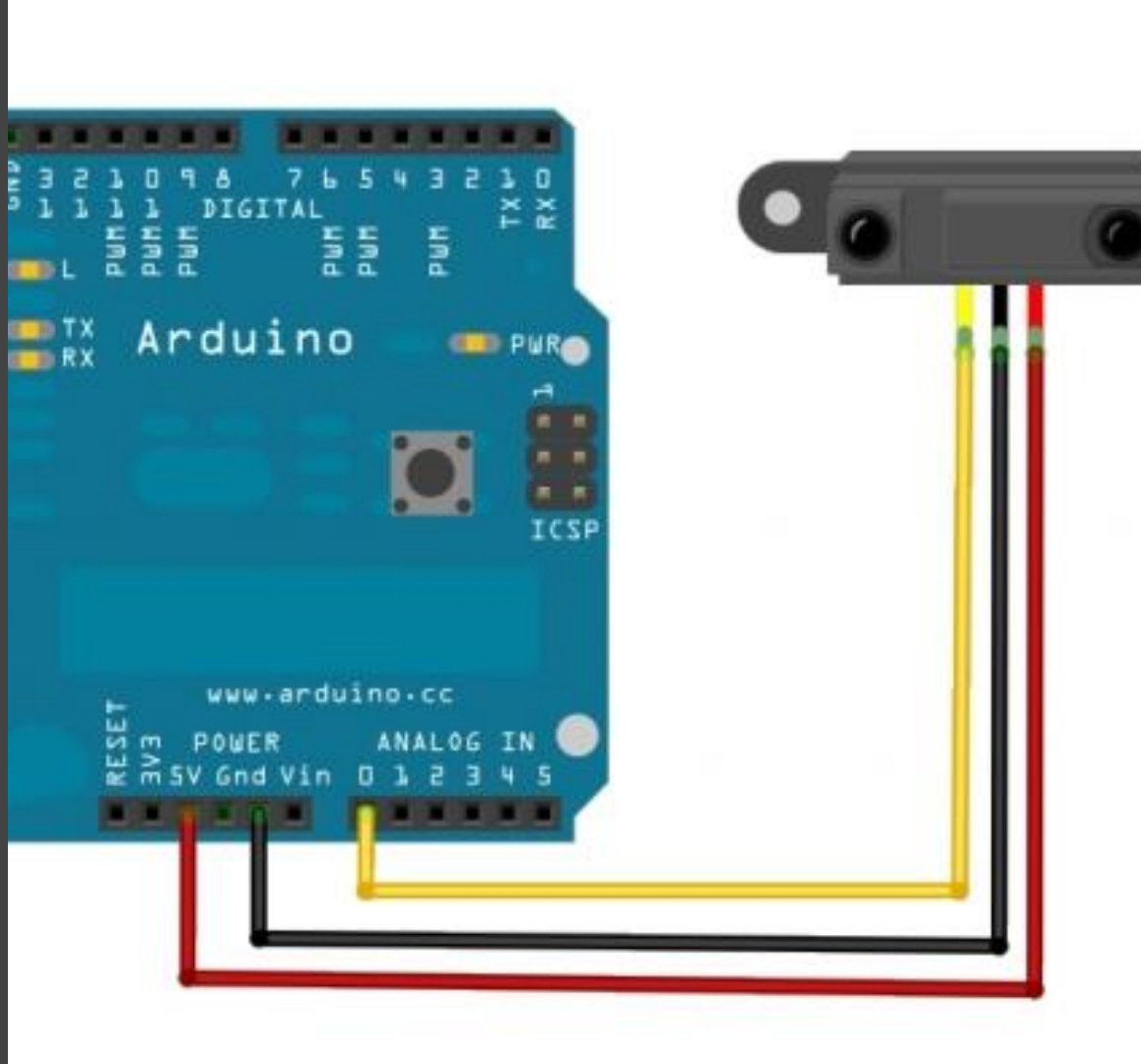
Joystick analog read

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(analogRead(0));
}
```



Infrared analog read

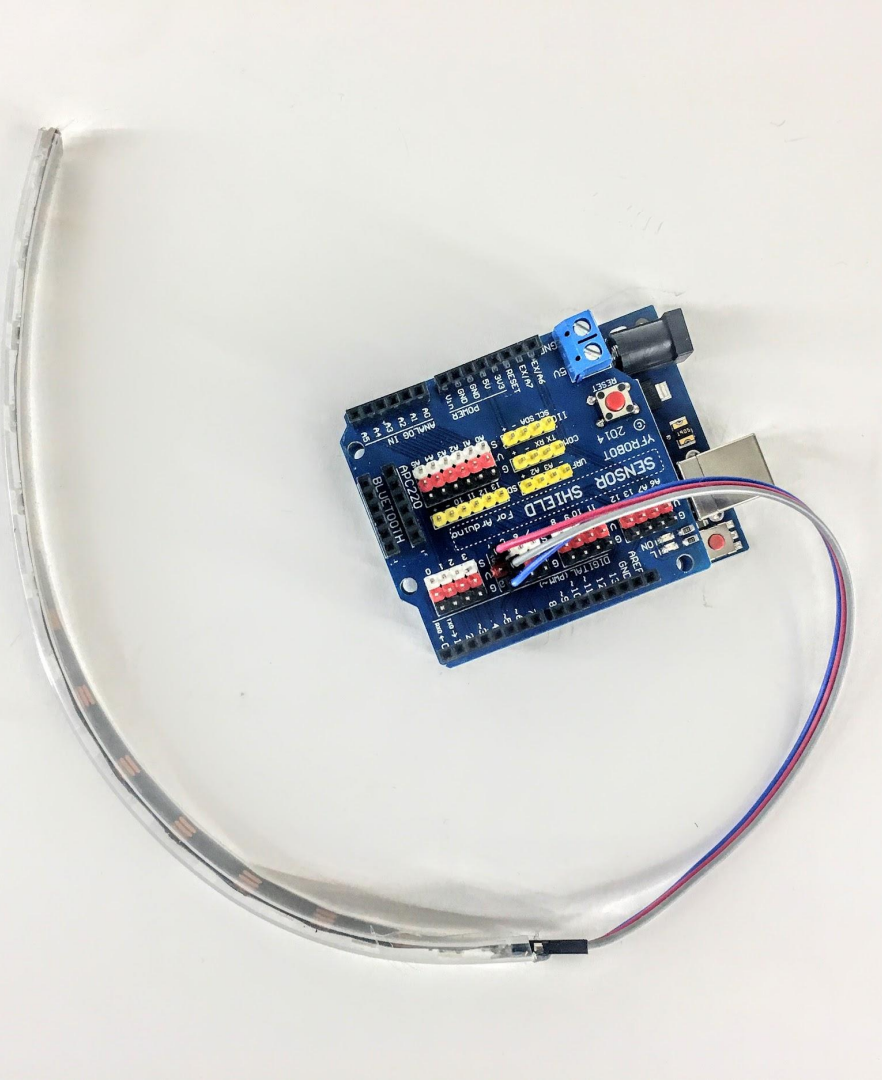
```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(analogRead(A0));
}
```



Output

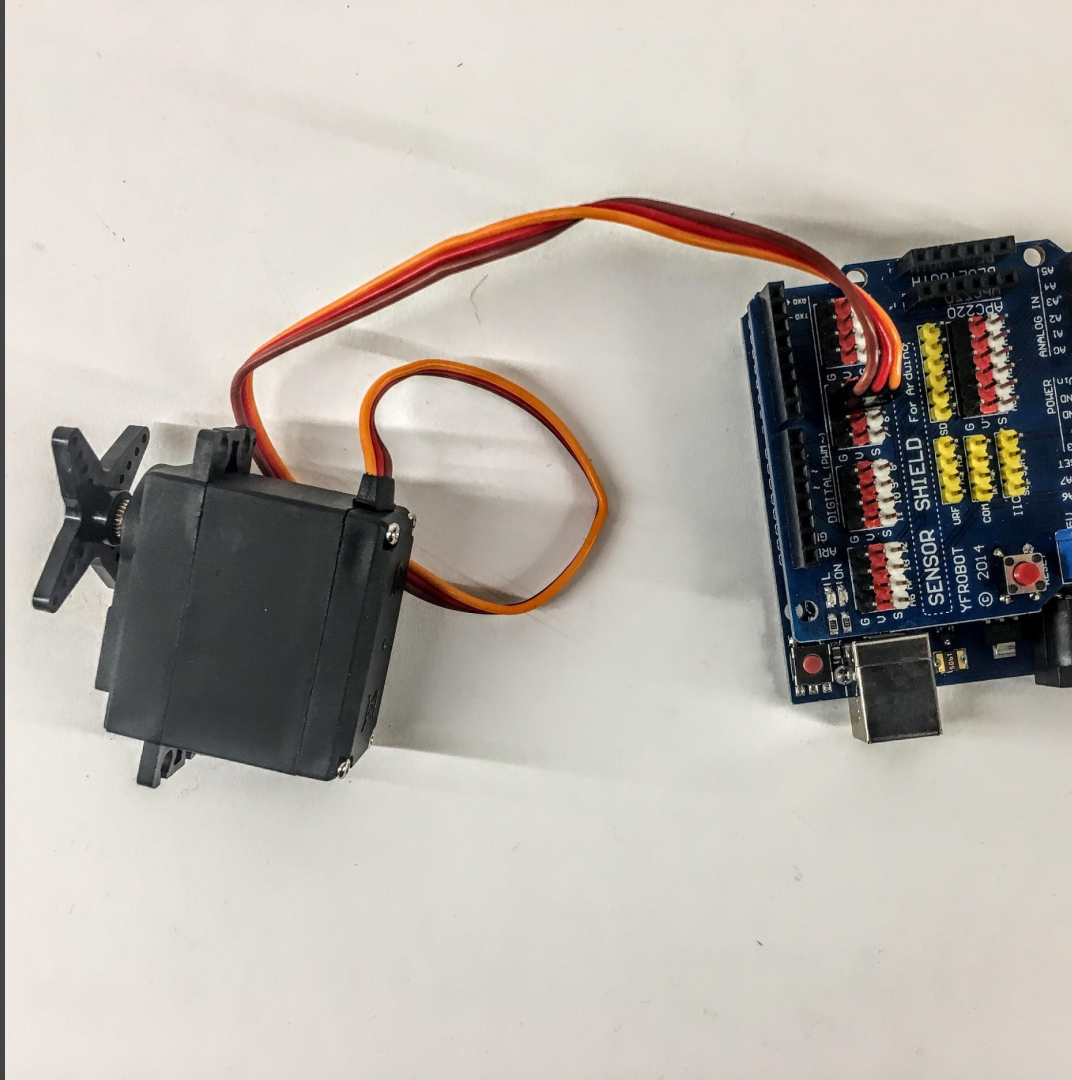
Button pressed

```
#include "FastLED.h"

// How many leds in your strip?
#define NUM_LEDS 10
#define DATA_PIN 4

// Define the array of leds
CRGB leds[NUM_LEDS];

void setup() {
        FastLED.addLeds<NEOPIXEL, DATA_PIN>(leds,
NUM_LEDS);
}

void loop() {
        // Turn the LED on, then pause
        leds[0]= CRGB( 255, 0, 0);
        leds[1]= CRGB( 0, 255, 0);
        leds[2]= CRGB( 0, 0, 255);
        FastLED.show();

}
```



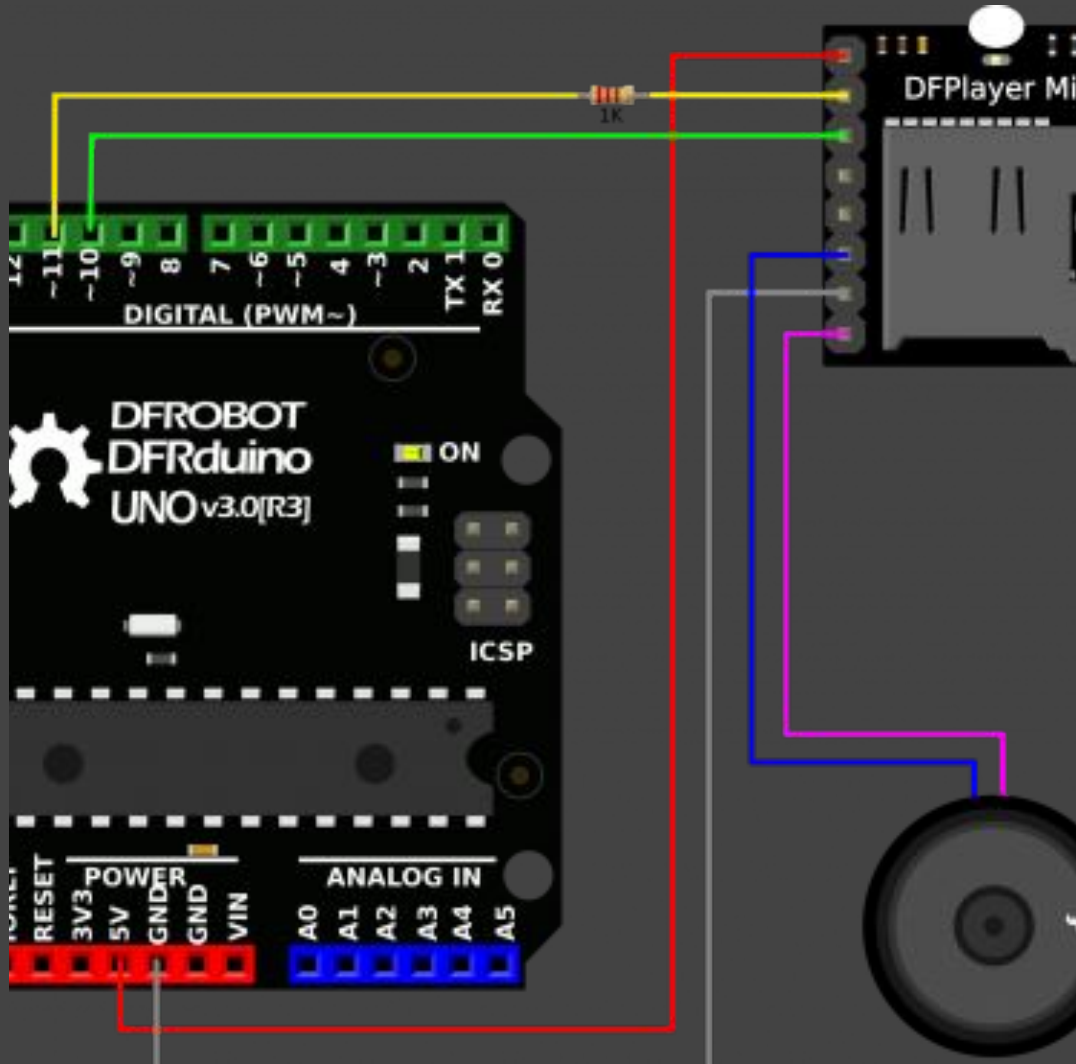
Control a servo

```
#include <Servo.h>

Servo myservo;
int pos = 0;

void setup() {
        myservo.attach(5);
 }

void loop() {
        pos = (pos +1)%160;
        myservo.write(pos);
 }
```



Play an mp3

```
#include <SoftwareSerial.h>
#include "Arduino.h"
#include <DFRobotDFPlayerMini.h>

SoftwareSerial mySoftwareSerial(10, 11); // RX, TX - mp3
DFRobotDFPlayerMini myDFPlayer;
boolean lastState = false;

void printDetail(uint8_t type, int value);
void setup() {
  mySoftwareSerial.begin(9600); //for mp3 player
  Serial.begin(9600);

  if (!myDFPlayer.begin(mySoftwareSerial)) {
//Use softwareSerial to communicate with mp3.
    Serial.println(F("Unable to begin:"));
    Serial.println(F("1.Please recheck the connection!"));
    Serial.println(F("2.Please insert the SD card!"));
    while (true);
  }
  myDFPlayer.volume(20); //Set volume value. From 0 to 30

  pinMode(3, INPUT_PULLUP);
}

void loop() {
  boolean btnPressed = !digitalRead(3);
  if (btnPressed == true && lastState == false)
  {
    myDFPlayer.play(1); //Play the first mp3
    Serial.println("button has been pressed.");
  }
  lastState = btnPressed;
}
```



Code examples

Rainbow pattern for one led:

```
int myColor = 100;

void freePattern()
{
        myColor = myColor +1;
        //restart: when mycolor hit 255 then set mycolor to zero
        if(myColor > 255)
        {
            myColor = 0;
        }
        leds[0] = CHSV(myColor,255,255);
}
```

Rainbow pattern for multiple leds led:

```
int myColor = 100;

void freePattern()
{
        if (myColor > 255)
          {
           myColor = 0;
          }

          for (int i = 0; i < NUM_LEDS-10; i++)
          {
           leds[i] = CHSV(myColor, 255, 255);
          }

          for (int i = NUM_LEDS-10; i < NUM_LEDS; i++)
          {
           leds[i] = CHSV(myColor, 255, 255);
          }

}
```

Simple chase pattern:

```
int theNextOne = 0;

void freePattern()
{
        if (millis() - timer2 > 2000) // do something every two seconds)
        {
                timer2 = millis();
                //do something here

                theNextOne = theNextOne + 1;
                if (theNextOne >= NUM_LEDS)
                {
                  theNextOne = 0;
                }
                // make the one behind black
                if (theNextOne != 0)
                {
                  leds[theNextOne - 1] = CRGB(0, 0, 0);
                }
                else
                {
                  leds[NUM_LEDS - 1] = CRGB(0, 0, 0);

                }


                leds[theNextOne] = CRGB(255, 0, 0);

        }


}
```

Combinations