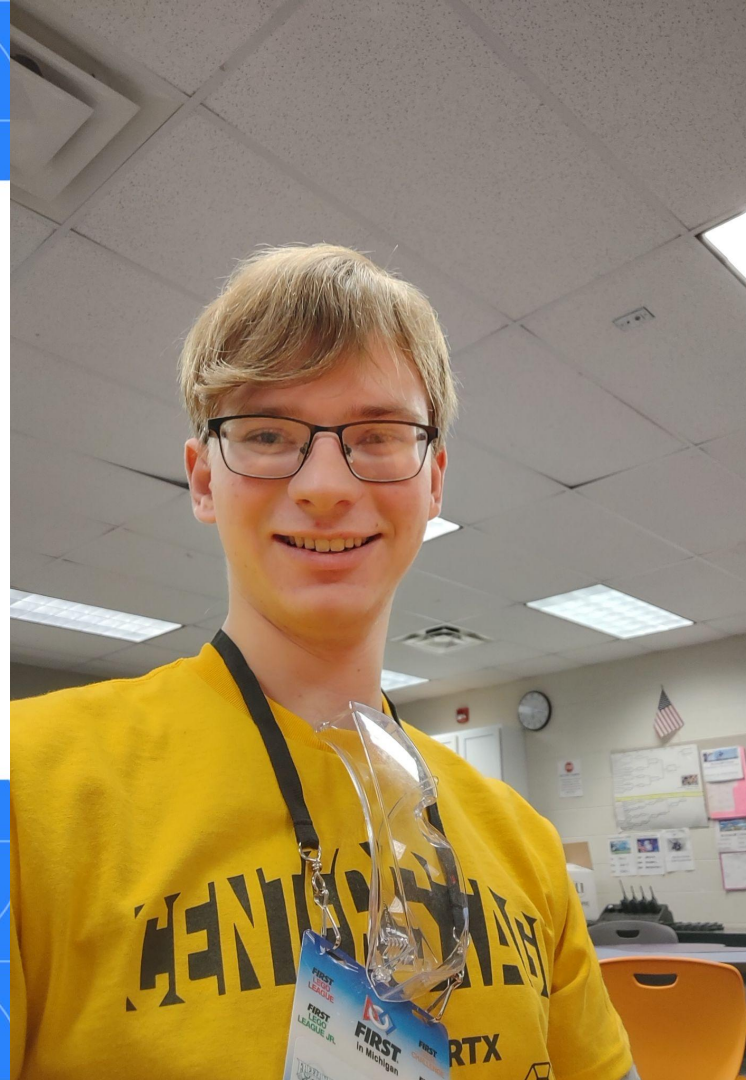
Google Developer Groups

Editable Location

Hardware Programming and Embedded Systems

Andrew Barton



Introduction

- Senior majoring in Electrical Engineering and Computer Engineering
- Originally from Minneapolis MN
- 9 years in robotics and multiple embedded systems internships
- I enjoy hiking, board games, and reading in my free time

Agenda

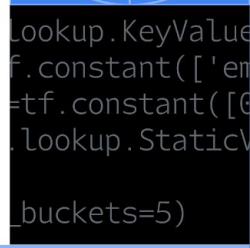
1. Introduction to Hardware Programming and Arduino
2. PWM
3. Control Systems

Parts List

Arduino Board	x1	
1kΩ Resistor	x2	(x3)
100µF Capacitor	x1	
LED	x2	(x3)
Potentiometer	x1	
HC-SR04 Ultrasonic Sensor	x1	
Servo Motor	x1	
Jumper Cables		

Google Developer Groups

What is Embedded Systems?



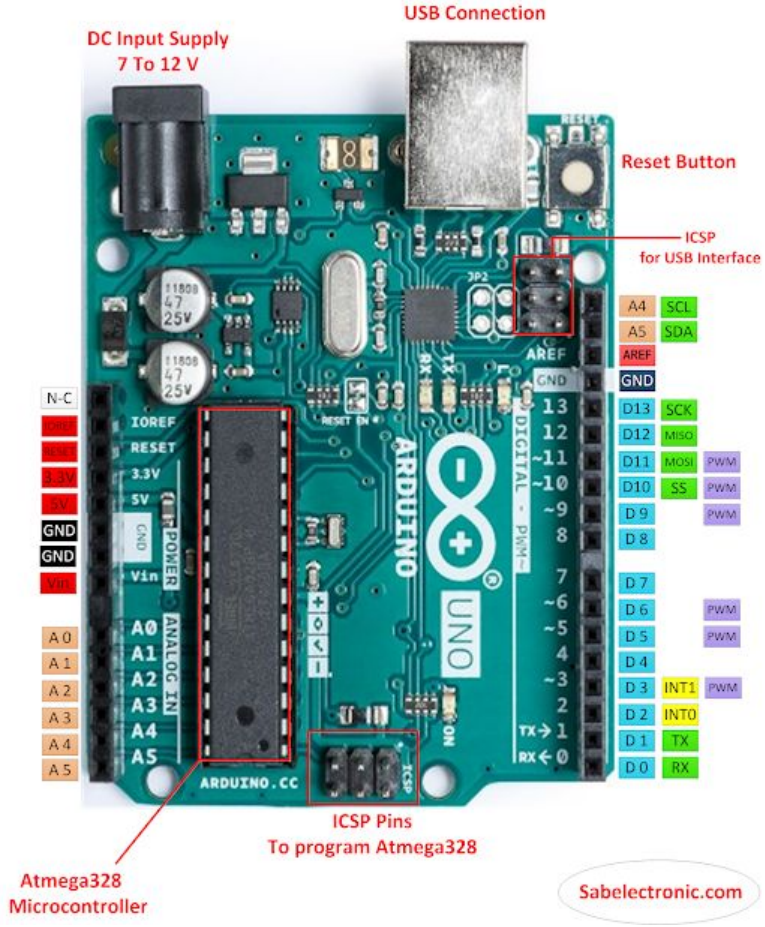
What is Embedded Systems?

Embedded systems is the term used to describe software implementation on hardware devices. This typically refers to devices designed to interact with the real world, such as sensors, input devices, and motor controls. Embedded systems consist of three main components:

1. Hardware device
2. Application software
3. Real time operating system

Arduino

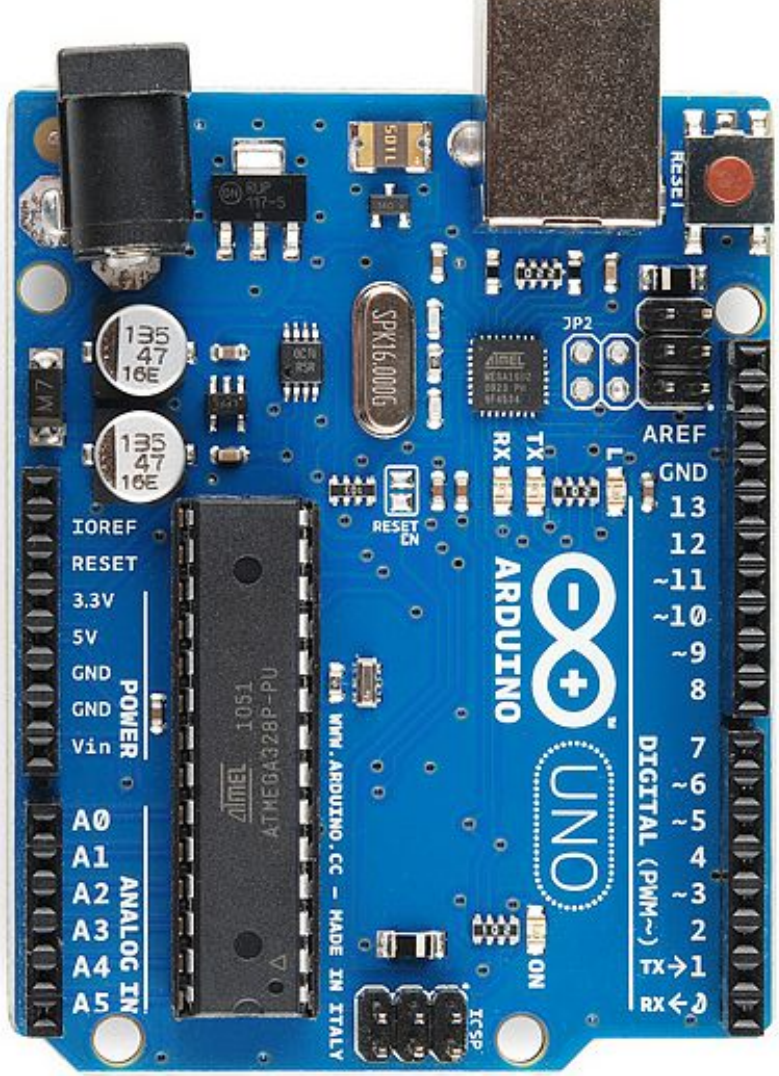
Arduino is a cheap and easy to use open source microcontroller platform. It is commonly used in electronics projects and can be easily configured for various applications.



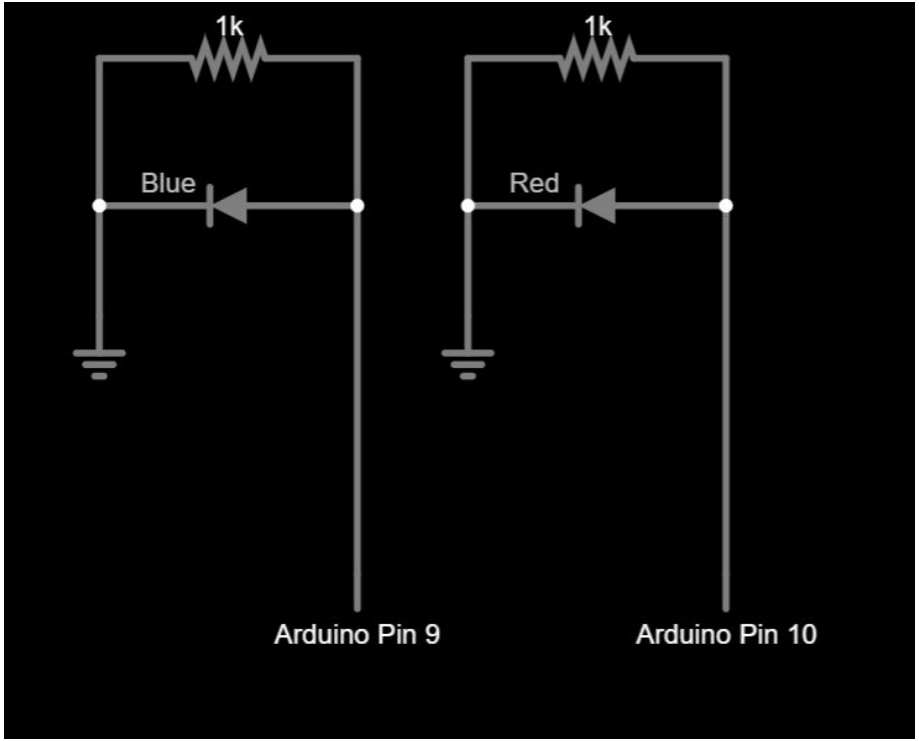
Arduino UNO Pinout

```
ldren: [
con(icon, color: color
ontainer(
 margin: const EdgeIns
 child: Text(
   label,
   style: TextS
```

Basic Arduino Example



Basic LED Circuit



Arduino Example

Setup:
Used to initialize pins and prepare to execute

Loop:
Runs continuously during program execution

FlashLED.ino

```
1   // init pins
2   uint8_t* port = &PORTH;
3   const uint8_t LEDPin1 = 6;
4   const uint8_t LEDPin2 = 10;
5
6   void setup() {
7     // put your setup code here, to run once:
8     // pinMode(LEDPin1, OUTPUT);
9     DDRH |= 1 << LEDPin1;
10    pinMode(LEDPin2, OUTPUT);
11  }
12
13  void loop() {
14    // put your main code here, to run repeatedly:
15
16    // read, modify, write example
17    *port |= 1 << LEDPin1;
18    delay(100);
19    *port &= ~(1 << LEDPin1);
20
21    // digital write example
22    digitalWrite(LEDPin2, HIGH);
23    delay(100);
24    digitalWrite(LEDPin2, LOW);
25
26  }
27
```

Arduino Example

Read, Modify, Write:
Perform binary operations to set pin values in port register

Takes bitwise OR of port values with a 1 left shifted to LED Pin 1 bit.
2_00000100

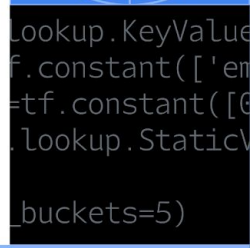
Takes bitwise AND of port values with inverse of 1 left shifted to LED Pin 1 bit.
2_11111011

FlashLED.ino

```
1   // init pins
2   uint8_t* port = &PORTH;
3   const uint8_t LEDPin1 = 6;
4   const uint8_t LEDPin2 = 10;
5
6   void setup() {
7     // put your setup co
8     // pinMode(LEDPin1, 
9     DDRH |= 1 << LEDPin1;
10    pinMode(LEDPin2, OUTP
11  }
12
13  void loop() {
14    // put your main code here, to run repeatedly:
15
16    // read, modify, write example
17    *port |= 1 << LEDPin1;
18    delay(100);
19    *port &= ~(1 << LEDPin1);
20
21    // digital write example
22    digitalWrite(LEDPin2, HIGH);
23    delay(100);
24    digitalWrite(LEDPin2, LOW);
25
26  }
27
```

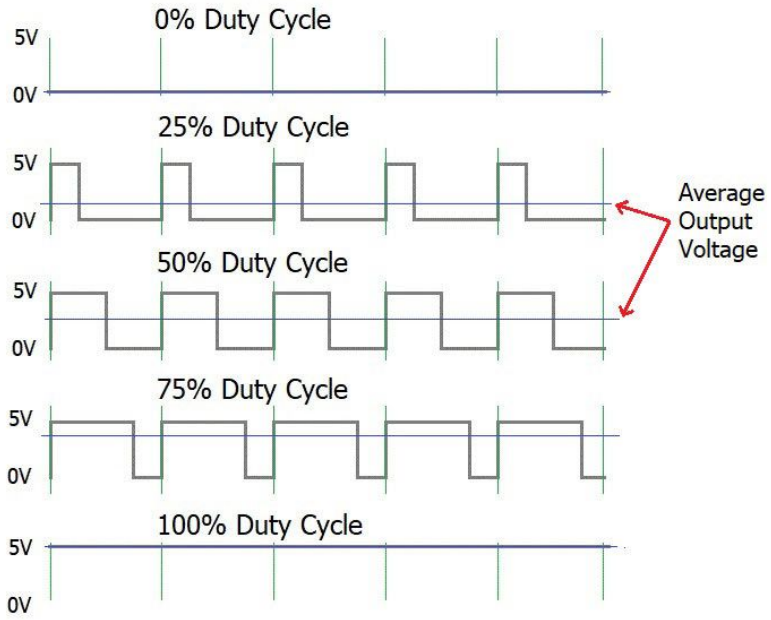
Google Developer Groups

Pulse Width Modulation

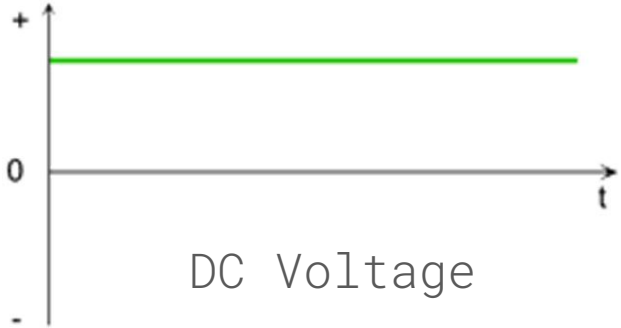


Pulse Width Modulation

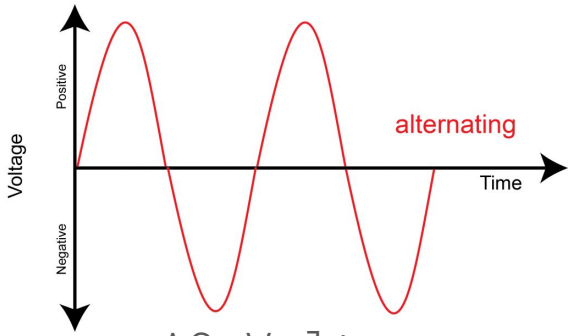
Pulse width modulation (PWM) is a control method to drop the voltage or power sent to a device through pulsing a DC voltage source to lower the average voltage delivered. The pulse width is called the duty cycle, and is expressed as a percentage of the total pulse and delay width.



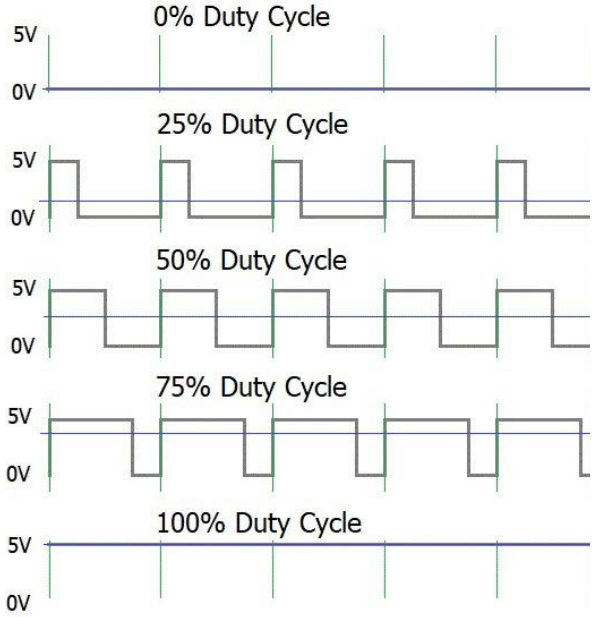
Pulse Width Modulation



DC Voltage



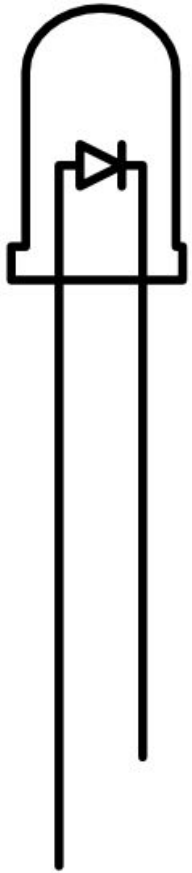
AC Voltage



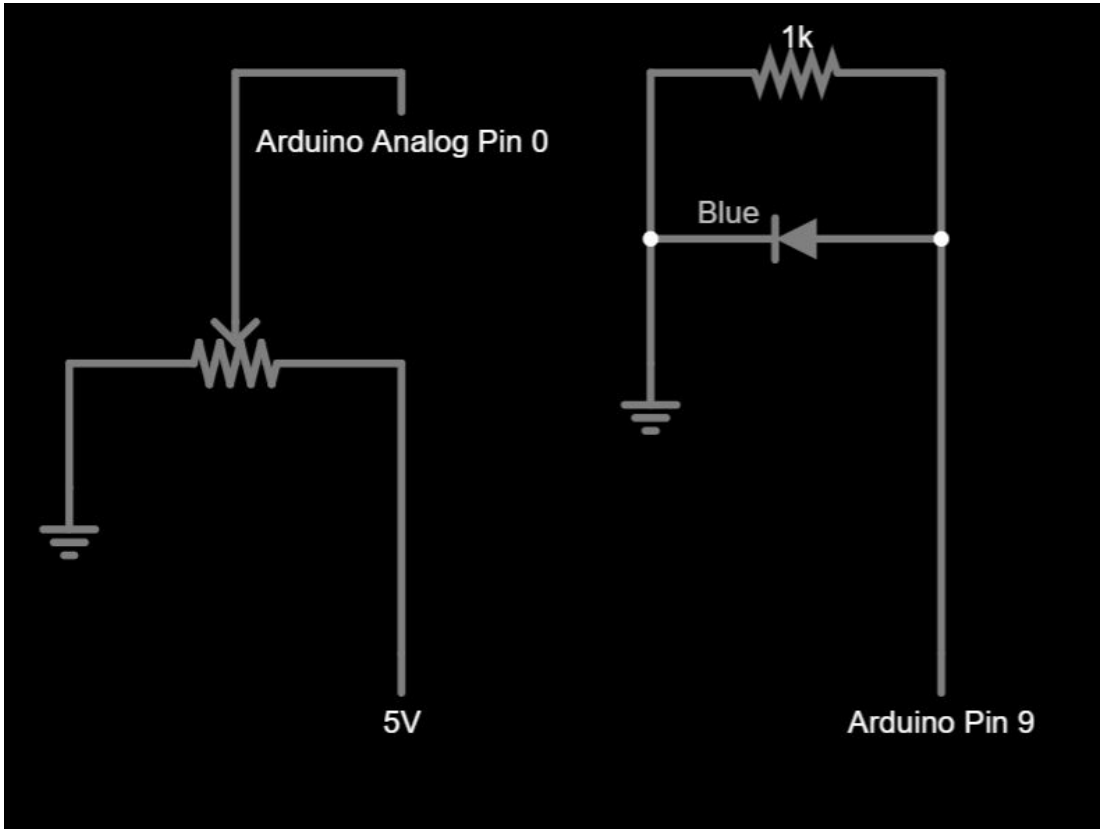
PWM Voltage

```
ldren: [
con(icon, color: color
ontainer(
 margin: const EdgeIns
 child: Text(
   label,
   style: TextS
```

PWM Example



PWM Circuit



PWM Example

PotLEDPWMUnfinished.ino

```
1   int potPin = A0;      // analog input pin A0
2   int ledPin = 9;       // digital output pin 9
3   int PWM = 0;  // current brightness of the LED
4   int time = 0;
5
6   void setup() {
7     pinMode(ledPin, OUTPUT);  // set the LED pin as output
8   }
9
10  void loop() {
11    // get value from pot
12
13    // set pin HIGH
14
15    // delay
16
17    // set pin LOW
18
19    // delay
20
21  }
22
```

PWM Example

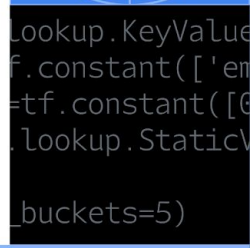
Duty cycle of PWM/1000%

PotLEDPWM.ino

```
1   int potPin = A0;      // analog input pin A0
2   int ledPin = 9;       // digital output pin 9
3   int PWM = 0;  // current brightness of the LED
4   int time = 0;
5
6   void setup() {
7     pinMode(ledPin, OUTPUT);  // set the LED pin as output
8   }
9
10  void loop() {
11    int potVal = analogRead(potPin);            // read the potentiometer value (0-1023)
12    PWM = map(potVal, 0, 1023, 0, 900);  // map the potentiometer value to 0-900
13
14    // PWM
15    digitalWrite(ledPin, HIGH);
16    delayMicroseconds(PWM);
17    digitalWrite(ledPin, LOW);
18    delayMicroseconds(1000 - PWM);
19
20  }
21
22
```

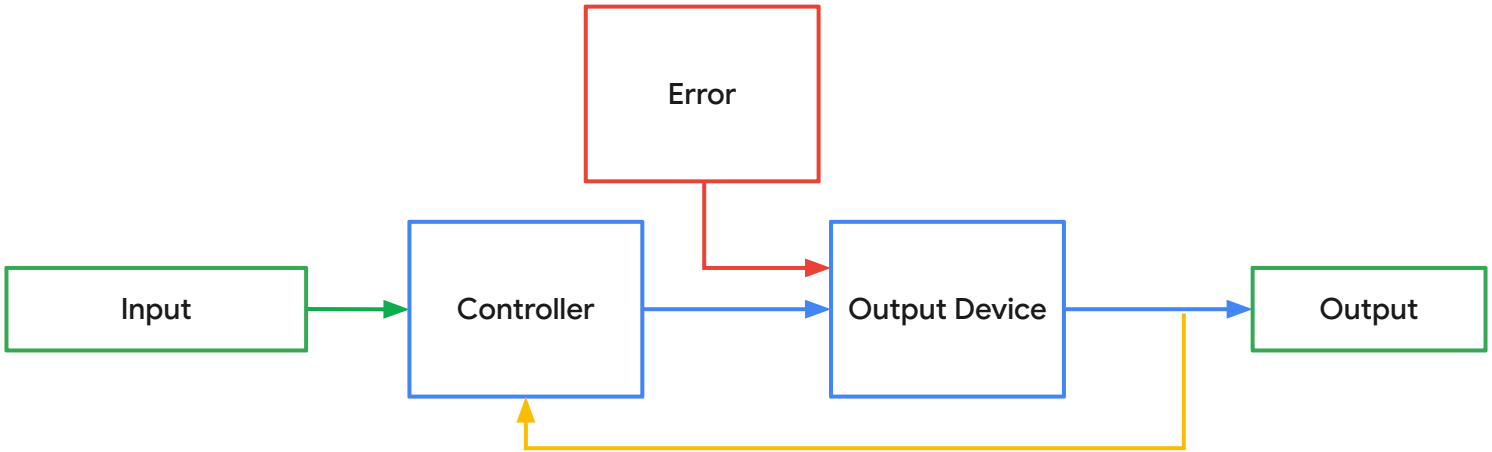
Google Developer Groups

Control Systems



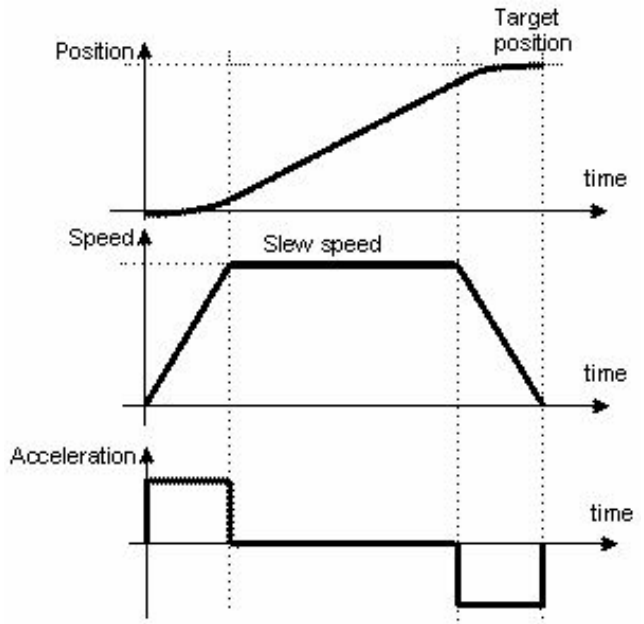
Control Loops

Control loops are feedback loops that detect error and implement corrective actions. Controls can be implemented in both software and hardware.



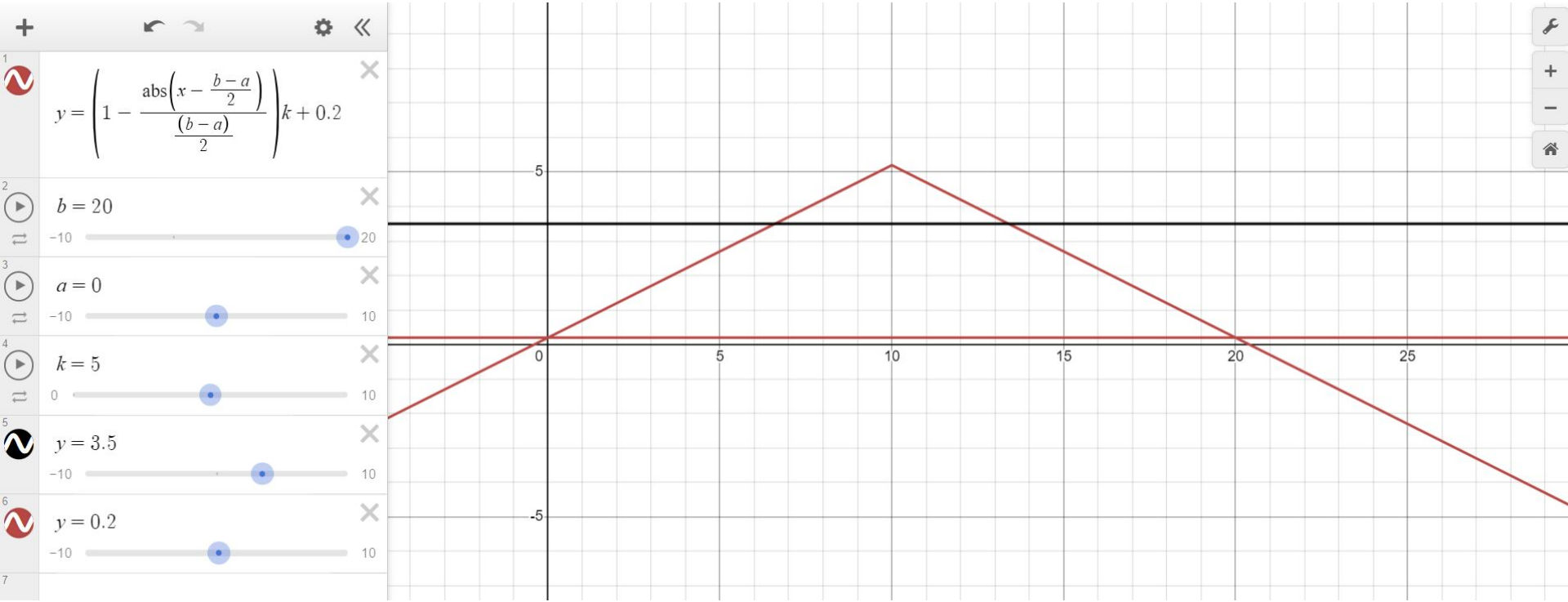
Trapezoidal Control

Trapezoidal control is one of the simplest control systems and does not implement any error correcting. Rather, it changes the intensity based on the system's proximity to completion. Because of this, it is not as accurate as other control methods.




```
trapazoidCtrl(start, end, maxSpeed,minSpeed, Kp) {
     currentPos = start;
     calcSpeed = minSpeed;
     midpoint = (end-start)/2
     while(currentPos < end) {
          dist = 1 - (abs(currentPos - midpoint) / midpoint);
          calcSpeed = calcSpeed + (dist * Kp) * sign(midpoint-currentPos);
          setMotorSpeed(min(calcSpeed, maxSpeed));
          currentPos = readEncoder();
     }
     setMotorSpeed(0);
}
```

Trapezoidal Control Derivation



Google Developer Groups

PID Control

PID control is one of the most common controllers and can be extremely accurate if tuned correctly. Tuning a PID controller involves tuning three variables in a single equation and is a long process. Each variable affects a different aspect of the control:

P - Controls magnitude of change when correcting error

I - Accounts for error over time, keeps heading in steady state

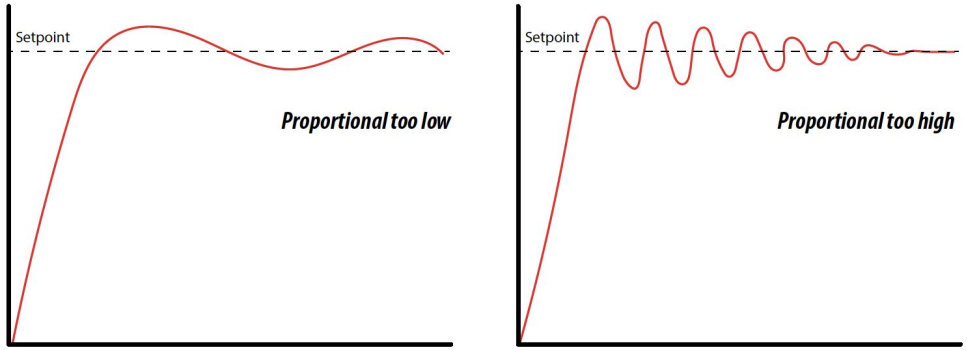
D - Limits speed of corrections

$$K_{\mathrm{p}}e + K_{\mathrm{i}}\int_0^t e(t)\,dt + K_{\mathrm{d}}\frac{de}{dt}$$

P I D

PID Control

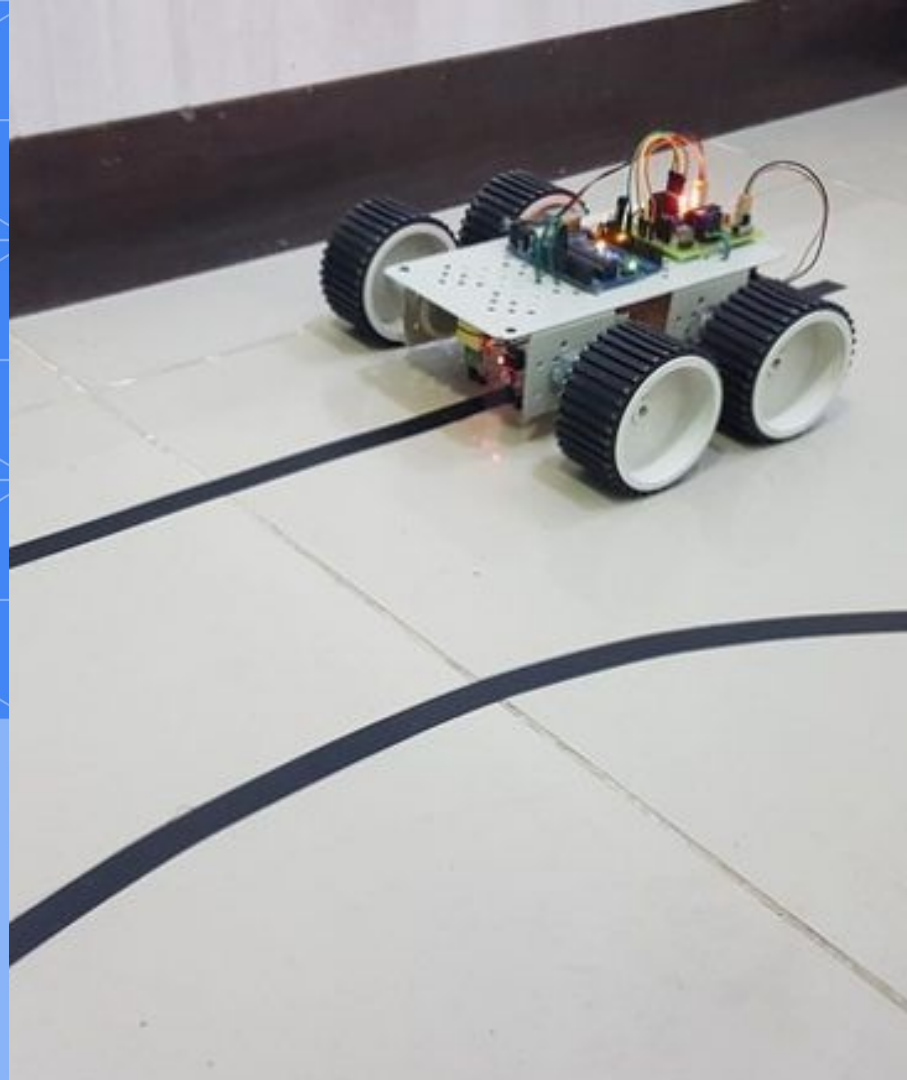
Proportional Control



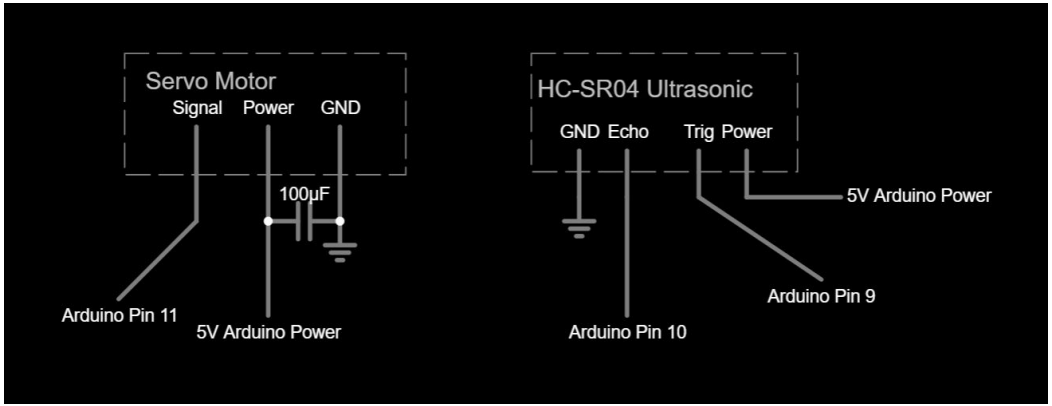
$$P_o = K_e(t) + p_0$$

```
ldren: [
con(icon, color: color
ontainer(
 margin: const EdgeIns
 child: Text(
   label,
   style: TextS
```

Proportional Control Example



Proportional Control Example



ServoProportionalControlUnfinished.ino

```
1   #include <Servo.h>
2
3   Servo rulerServo;
4
5   const int trigPin = 9;
6   const int echoPin = 10;
7
8   float duration, distance;
9   float pos = 45;
10
11  void setup() {
12    pinMode(trigPin, OUTPUT);
13    pinMode(echoPin, INPUT);
14    rulerServo.write(pos);
15    rulerServo.attach(11);
16    Serial.begin(9600);
17  }
18
19  void loop() {
20
21  }
22
```

Proportional Control Example

ServoProportionalControl.ino

```
#include <Servo.h>

Servo rulerServo;

const int trigPin = 9;
const int echoPin = 10;

float duration, distance;
float targetDist = 10; // cm
float Kp = 1;
float error;
float pos = 45;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  rulerServo.write(pos);
  rulerServo.attach(11);
  Serial.begin(9600);
}
```

```
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = (duration*.0343)/2;

  error = distance - targetDist;
  pos += error*Kp;
  if(pos > 160){
    pos = 160;
  } else if(pos < 20){
    pos = 20;
  }
  rulerServo.write(pos);

  Serial.print("Error: ");
  Serial.println(error);
  delay(1000);

  if(abs(error) < 0.3){
    Serial.print("Terminated. Final Error: ");
    Serial.println(error);
    delay(100);
    exit(0);
  }

}
```

Proportional Control Example

Pulse ultrasonic sensor

Get distance by multiplying by speed of sound in cm/uS

Calculate error and new position

ServoProportionalControl.ino

```
#include <Servo.h>

Servo rulerServo;

const int trigPin = 9;
const int echoPin = 10;

float duration, distance;
float targetDist = 10; // cm
float Kp = 1;
float error;
float pos = 45;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  rulerServo.write(pos);
  rulerServo.attach(11);
  Serial.begin(9600);
}
```

```
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = (duration*.0343)/2;

  error = distance - targetDist;
  pos += error*Kp;
  if(pos > 160){
    pos = 160;
  } else if(pos < 20){
    pos = 20;
  }
  rulerServo.write(pos);

  Serial.print("Error: ");
  Serial.println(error);
  delay(1000);

  if(abs(error) < 0.3){
    Serial.print("Terminated. Final Error: ");
    Serial.println(error);
    delay(100);
    exit(0);
  }

}
```

Google Developer Groups

Sources

https://www.tutorialspoint.com/embedded_systems/es_overview.htm

https://www.ninniku.tw/shine-bright-with-potentiometer-controlling-led-brightness-with-pwm-on-arduino/

https://projecthub.arduino.cc/Isaac100/getting-started-with-the-hc-sr04-ultrasonic-sensor-7cabe1

https://robotics.stackexchange.com/questions/9786/how-do-the-pid-parameters-kp-ki-and-kd-affect-the-heading-of-a-differential