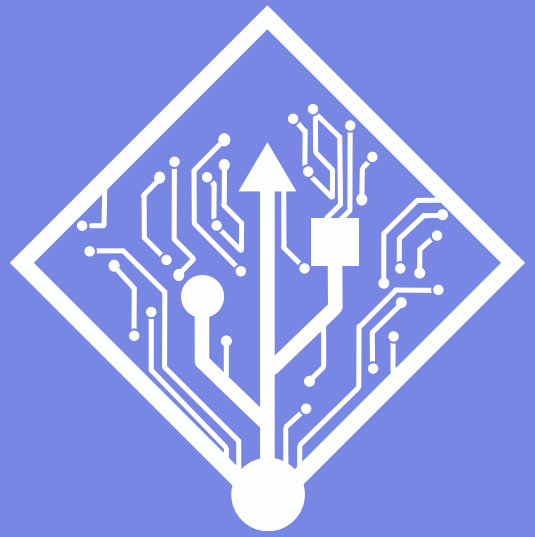


IEEE CITY

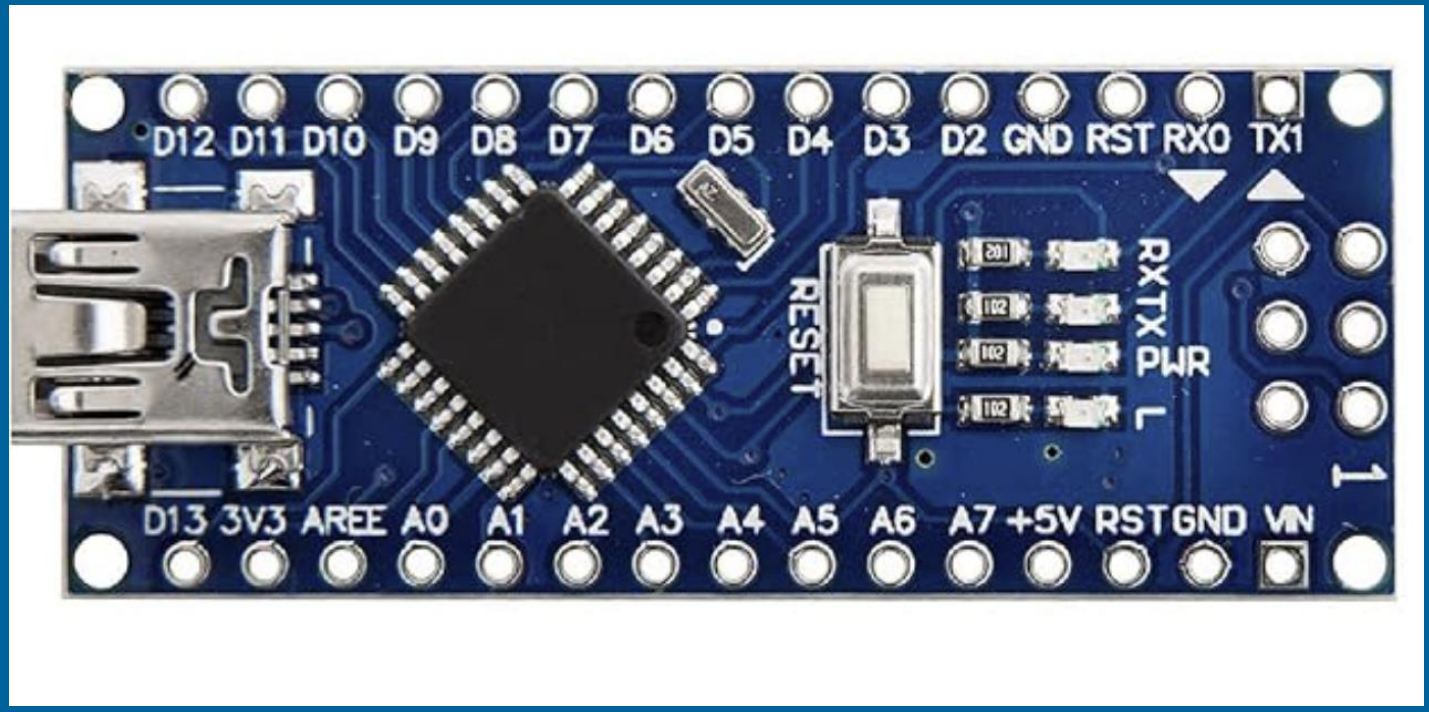
ROBOTICS SOCIETY

PROGRAMMING 1: MOTORS AND ENCODERS

IEEE

ARDUINO RECAP

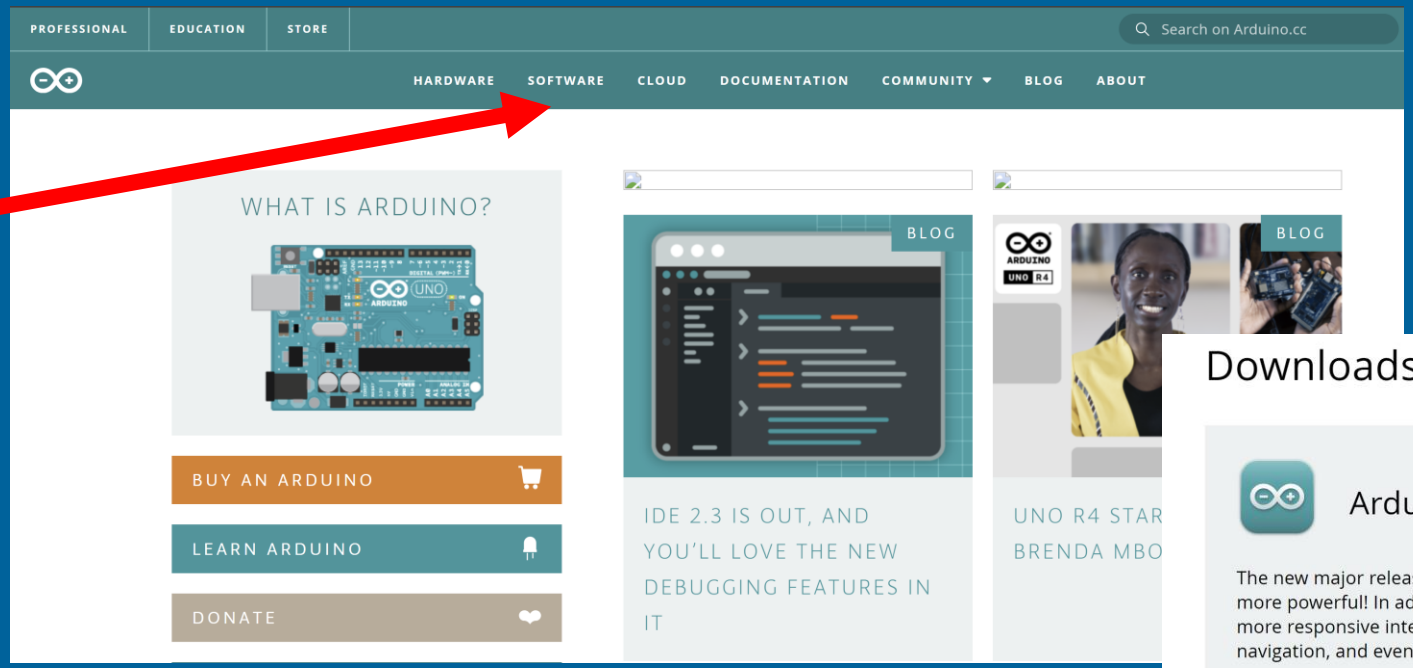
- 14 digital pins D0-D12 – Takes LOW or HIGH value
- 8 Analog input A0-A7 – Reads values from 0-1023
- 6 PWM Output – Writes values from 0-255
- 2 Interrupt pins

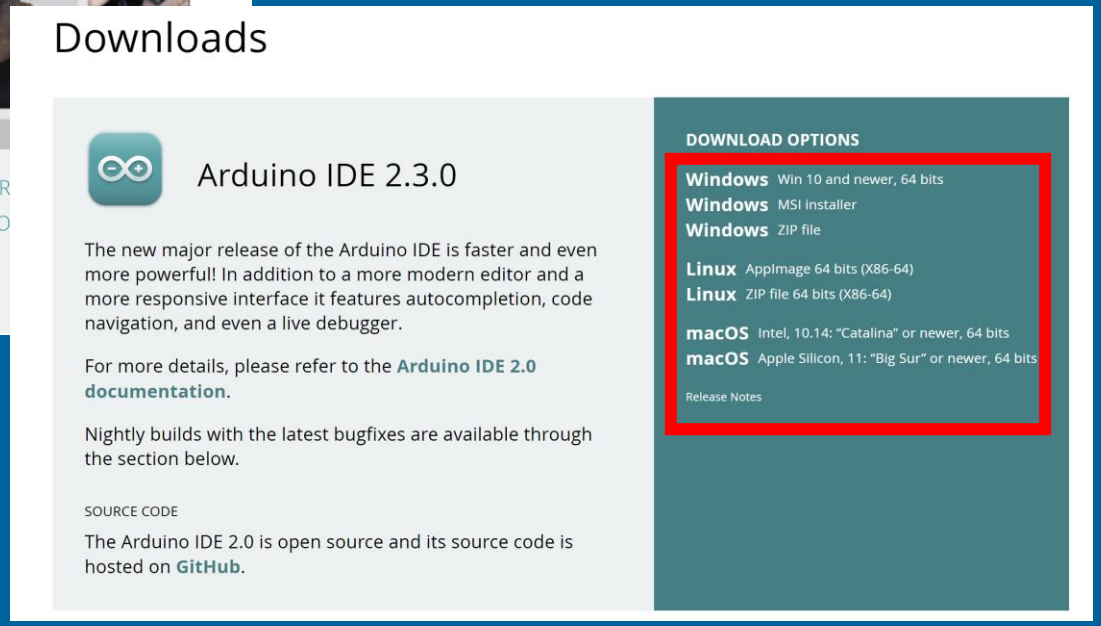


IEEE

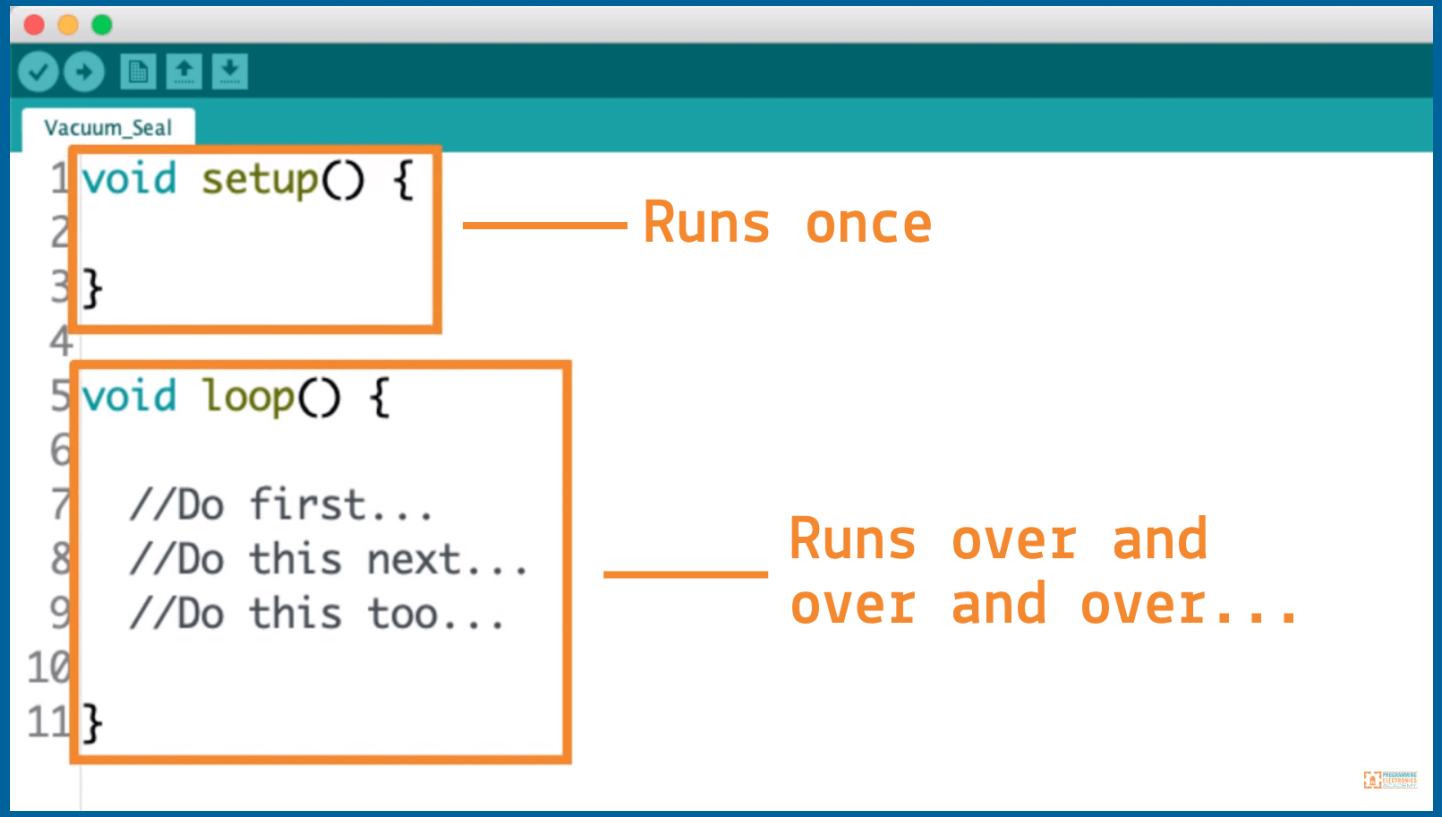
IDE INSTALLATION

Arduino.cc





SETUP AND LOOP ()



KEY ARDUINO FUNCTIONS

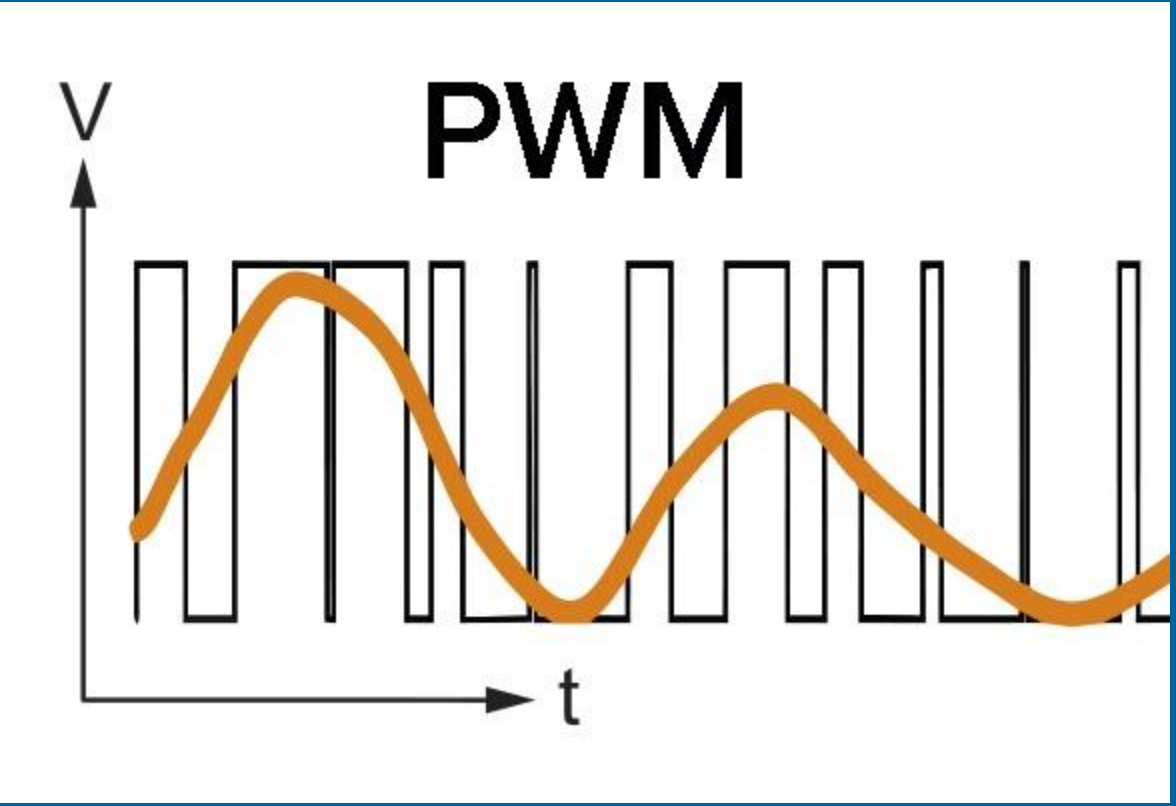
- PinMode(Pin, Mode) – Sets a pin to be input or output

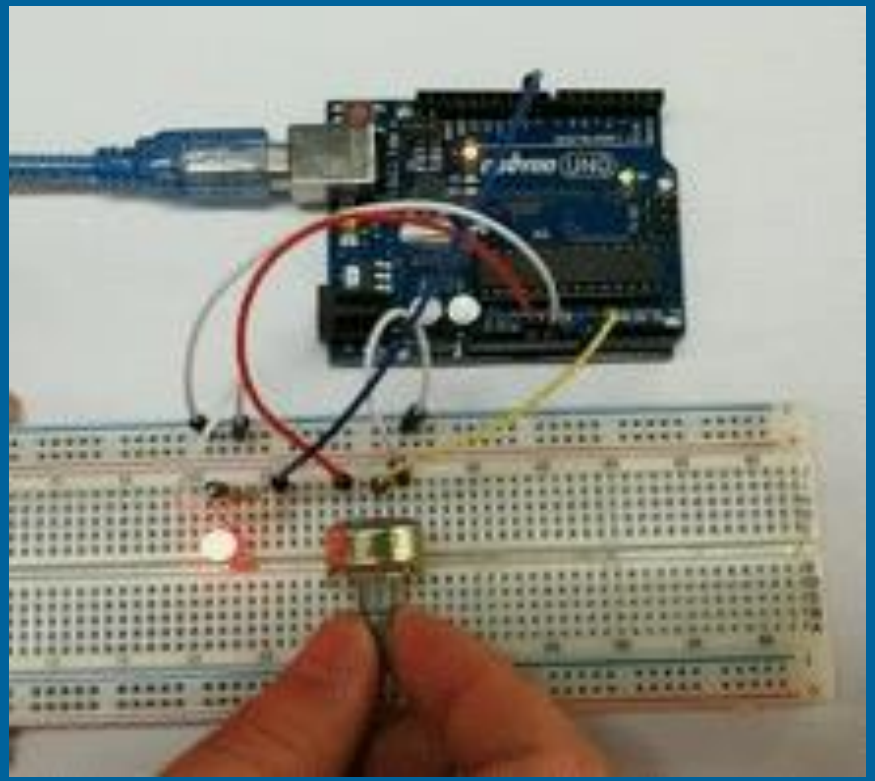
- DigitalRead(Pin) – Reads digital values
- DigitalWrite(Pin, Value) – Writes digital values

- AnalogRead(Pin) – reads analog values 0-1023
- AnalogWrite(Pin, Value) writes analog values 0-255

- Serial.print() – prints out things to the serial monitor

PWM RECAP





Components: N20 Motor

- **DIRECTION:**
  Clockwise = HIGH, Anticlockwise = LOW
  SPEED:
  PWM value analogWrite between 0-255

- Left motor direction pin: D7
  Left motor speed: D9

- Right motor direction pin: D8
- Right motor speed pin: D10

Gear Ratio 20:1

Setting up our code

```
const int SPEED_MOTOR_L = 9; // PWM MOTOR LEFT
const int SPEED_MOTOR_R = 10; // PWM MOTOR RIGHT

const int DIR_MOTOR_L = 7; // DIRECTION MOTOR LEFT
const int DIR_MOTOR_R = 8; // DIRECTION MOTOR RIGHT
```

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

  pinMode(SPEED_MOTOR_L, OUTPUT);
  pinMode(SPEED_MOTOR_R, OUTPUT);
  pinMode(DIR_MOTOR_L, OUTPUT);
  pinMode(DIR_MOTOR_R, OUTPUT);
```

Testing our motors

```
void loop(){

  digitalWrite(DIR_MOTOR_L, HIGH);
  analogWrite(SPEED_MOTOR_L, 150);

  digitalWrite(DIR_MOTOR_R, HIGH);
  analogWrite(SPEED_MOTOR_R, 150);

}
```

Spot any errors?

Testing our motors

```
void loop(){

  digitalWrite(DIR_MOTOR_L, HIGH);
  analogWrite(SPEED_MOTOR_L, 150);

  digitalWrite(DIR_MOTOR_R, HIGH);
  analogWrite(SPEED_MOTOR_R, 150);

}
```

Don’t forget one motor is the other way round!!

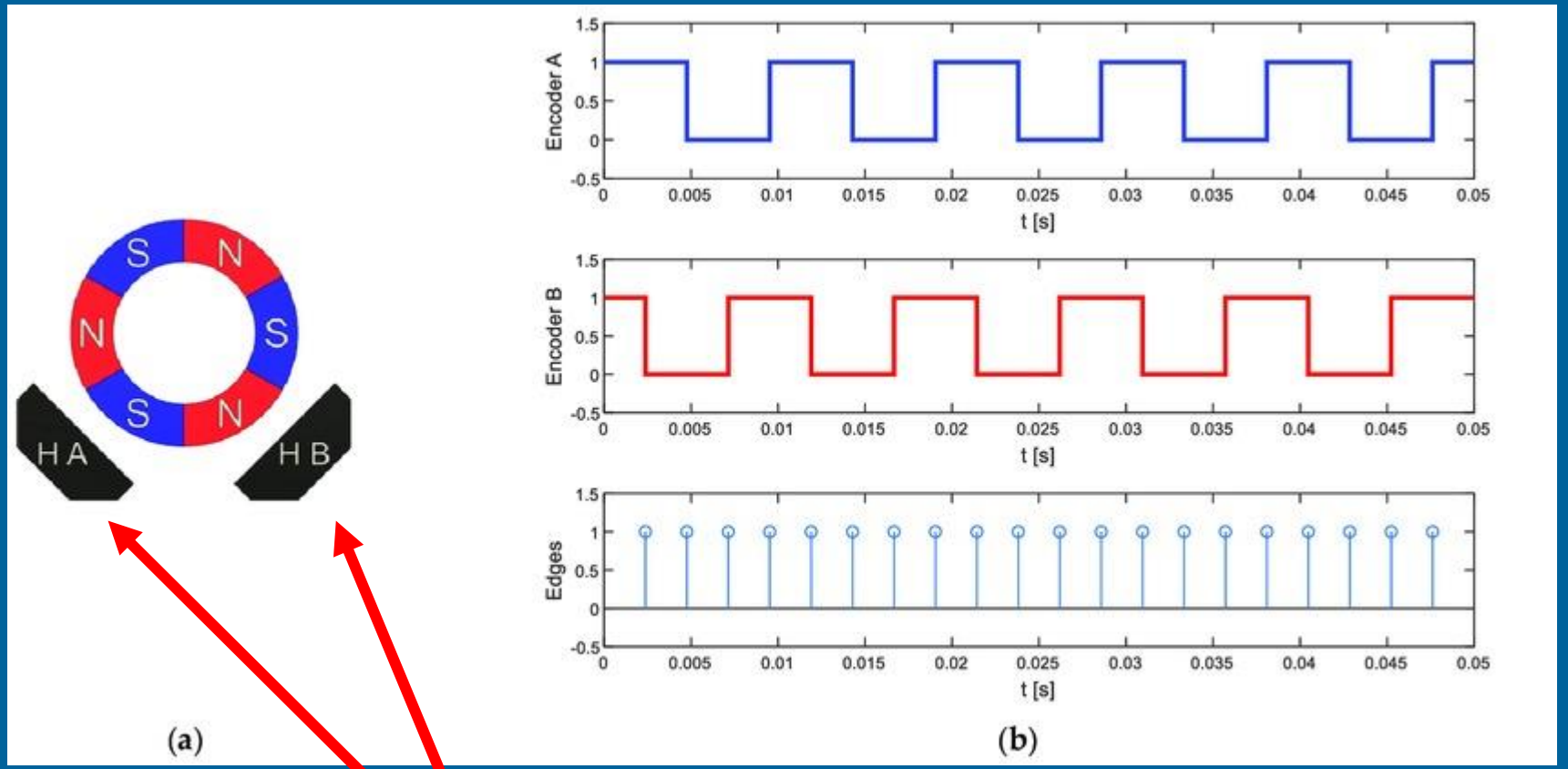
IEEE

Creating a function to control both motors for us

```
149  void setMotors(int dir, int speed){
150    analogWrite(SPEED_MOTOR_L, speed);
151    analogWrite(SPEED_MOTOR_R, speed);
152
153    if(dir == 1){
154      fast_write_pin(DIR_MOTOR_L, HIGH);
155      fast_write_pin(DIR_MOTOR_R, LOW);
156    } else if (dir == -1){
157      fast_write_pin(DIR_MOTOR_L, LOW);
158      fast_write_pin(DIR_MOTOR_R, HIGH);
159    } else{
160      analogWrite(SPEED_MOTOR_L, 0);
161      analogWrite(SPEED_MOTOR_R, 0);
162    }
163  }
```

How do we calculate distance??

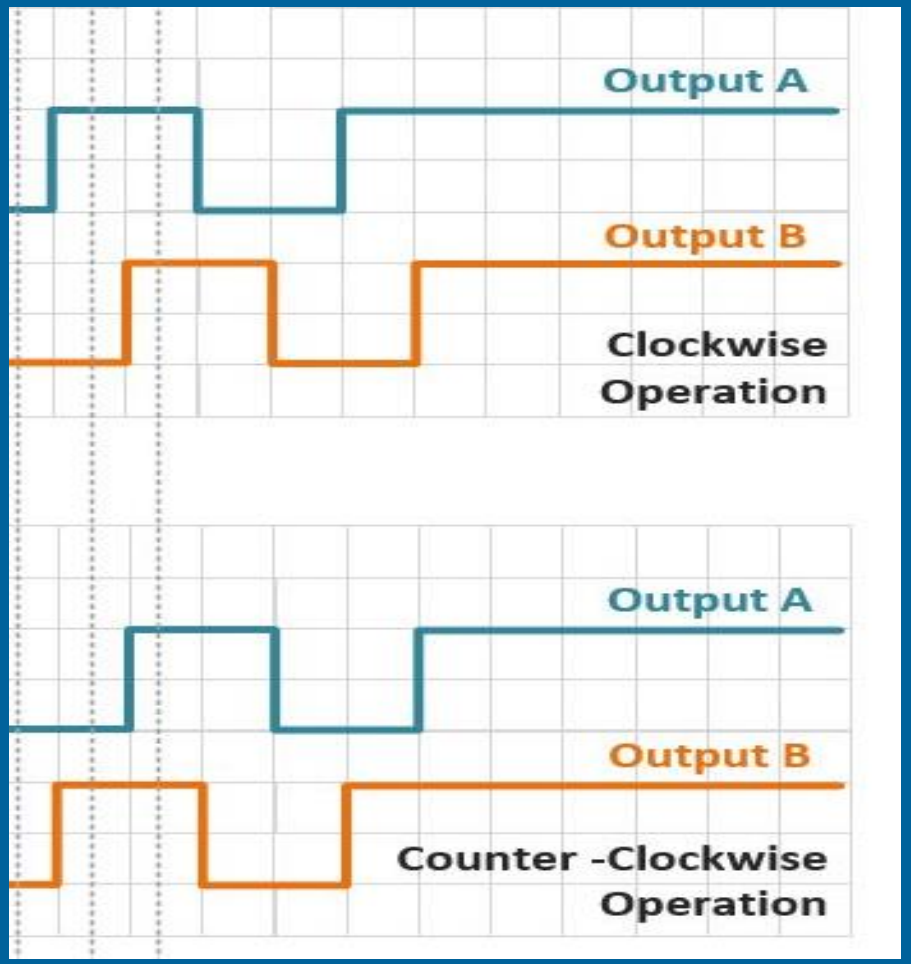
Components: Magnetic Encoder



What function would read this?

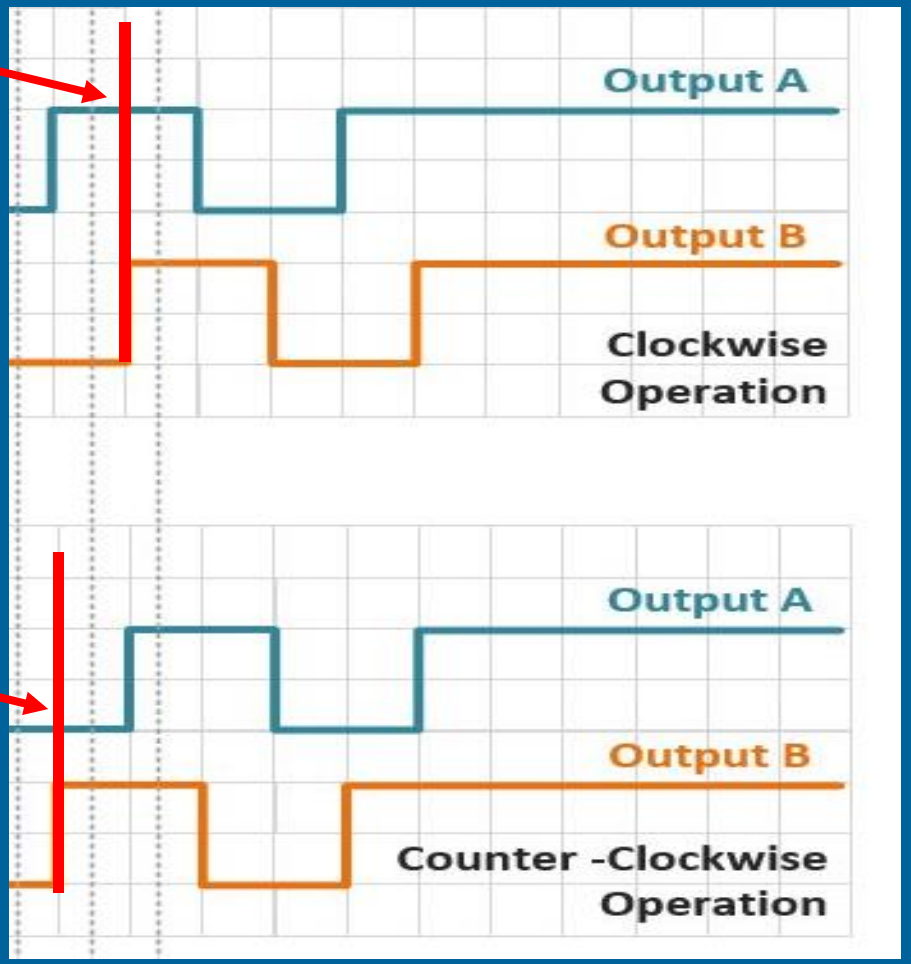
IEEE

Counting encoder pulses



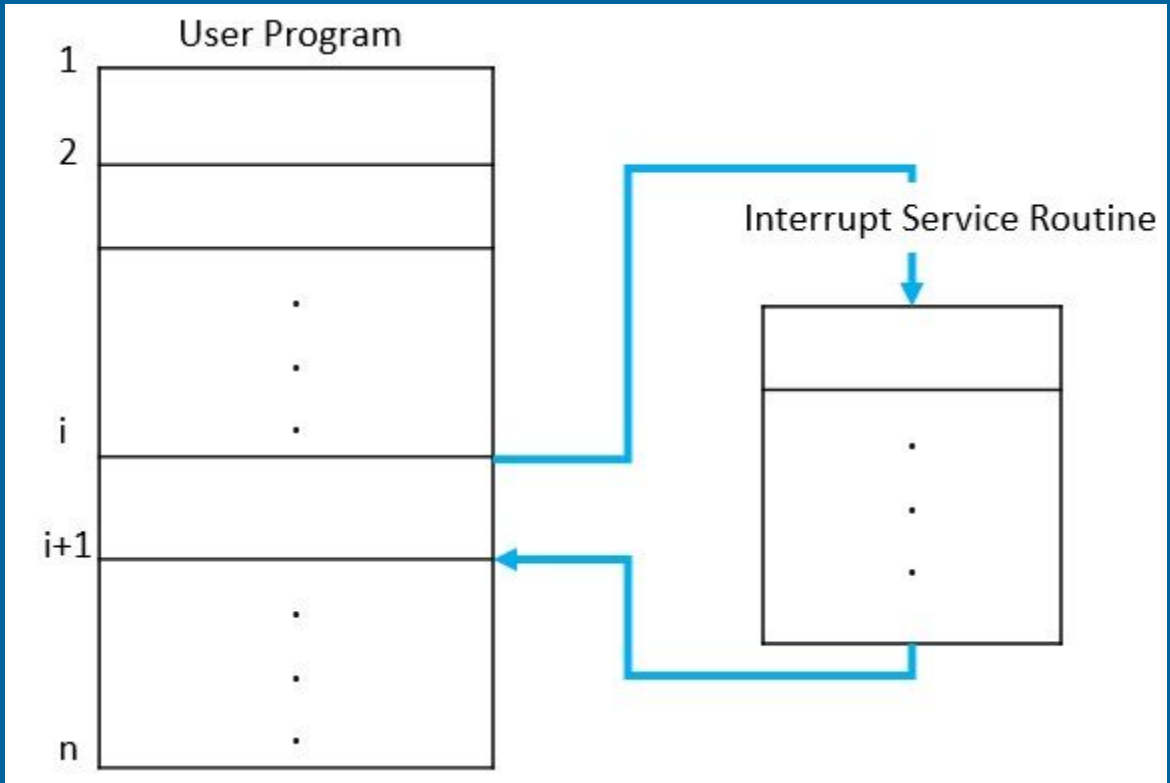
Counting encoder pulses

- A is HIGH



- A is LOW

What is an Interrupt?



- Arduino has a special function: attachInterrupt() to use in setup

Setting up our code

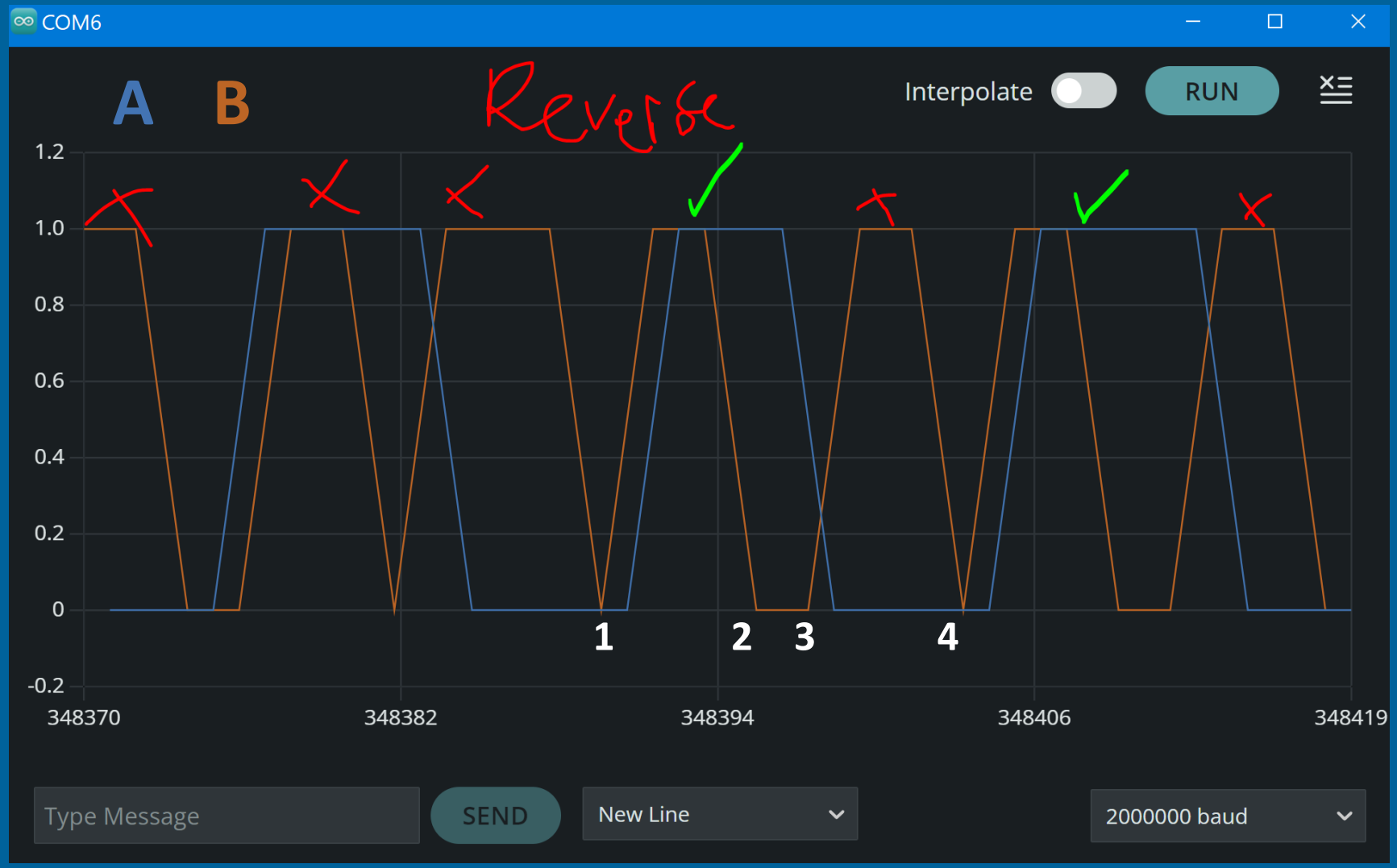
```
const int ENCODER_R_A = 3; // ENCODER RIGHT A (ticks first when motor forward)
const int ENCODER_R_B = 5; // ENCODER RIGHT B (ticks first when motor backward)

const int ENCODER_L_A = 4; // ENCODER LEFT A (ticks first when motor forward)
const int ENCODER_L_B = 2; // ENCODER LEFT B (ticks first when motor backward)

void setup() {
  ...
  pinMode(ENCODER_R_A, INPUT_PULLUP);
  pinMode(ENCODER_R_B, INPUT_PULLUP);
  pinMode(ENCODER_L_A, INPUT_PULLUP);
  pinMode(ENCODER_L_B, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(ENCODER_L_B), readEncoder, RISING);
```

Counting encoder pulses: Code

```
volatile int encoderCount = 0;
void readEncoder(){
  if(digitalRead(ENCODER_L_A) == HIGH){
    encoderCount++;
  } else{
    encoderCount--;
  }
}
```

Counting encoder pulses: Reality

Counting encoder pulses: Reality



Counting encoder pulses: Code Reality

```
98  void interruptHandlerLeft() {
99    if (interruptOccurred) {
100     if(uptick == 4){
101       endTime = micros();
102       uptick = 1;
103     }
104     if(uptick == 3 && fast_read_pin(ENCODER_L_A) == LOW && isActive == true){ // If A ac
105           leftEncoderPos++;
106           isActive = false;
107       } else if(uptick == 2 && fast_read_pin(ENCODER_L_A) == HIGH && isActive == false){
108           leftEncoderPos--;
109       }
110   } else {
111     if(uptick == 1){
112       startTime = micros();
113       if(fast_read_pin(ENCODER_L_A) == HIGH){
114         isActive = true;
115       } else{
116         isActive = false;
117       }
118     }
119     uptick++;
120   }
121   interruptOccurred = !interruptOccurred;
122 }
```

Counting encoder pulses: Code Reality

```
79  void readEncoderLeft() {
80    static uint8_t prevState = 0;
81    static uint8_t currState = 0;
82    static unsigned long lastTime = 0;
83
84    currState = (fast_read_pin(ENCODER_L_B) << 1) | fast_read_pin(ENCODER_L_A);
85
86    unsigned long currentTime = micros();
87    unsigned long deltaTime = currentTime - lastTime;
88    lastTime = currentTime;
89
90    // direction based on prev state
91    uint8_t direction = (prevState << 2) | currState;
92    switch(direction) {
93      case 0b0001:
94      case 0b0111:
95      case 0b1110:
96      case 0b1000:
97        leftEncoderPos++;
98        break;
99      case 0b0010:
100     case 0b1100:
101     case 0b0101:
102     case 0b1011:
103       leftEncoderPos--;
104       break;
105
106     default:
107       break;
108   }
```

- http://www.buxtronix.net/2011/10/rotary-encoders-done-properly.html

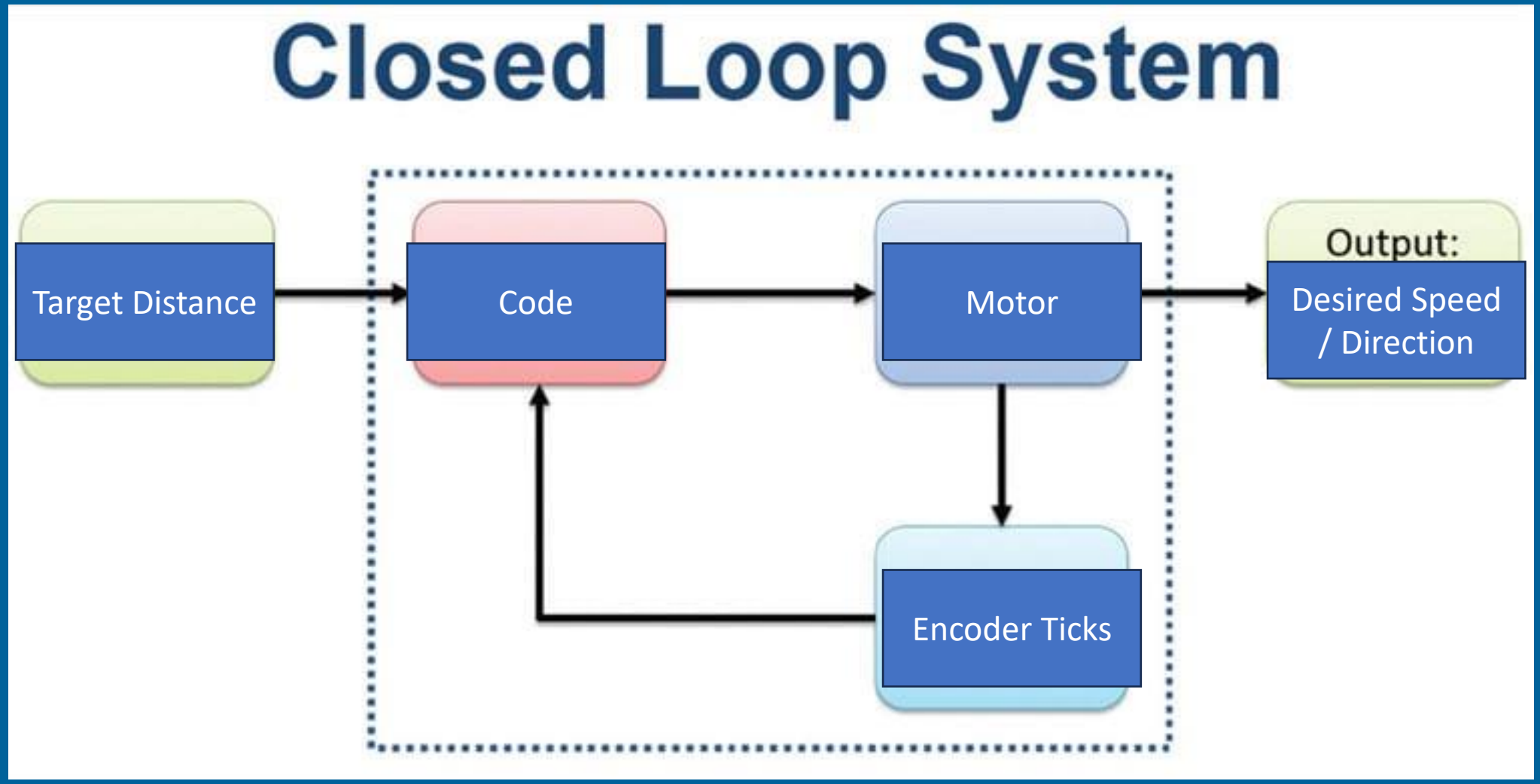
How do we make our robot move specified distances

- Friction?
- Inconsistent motor eg left quicker?

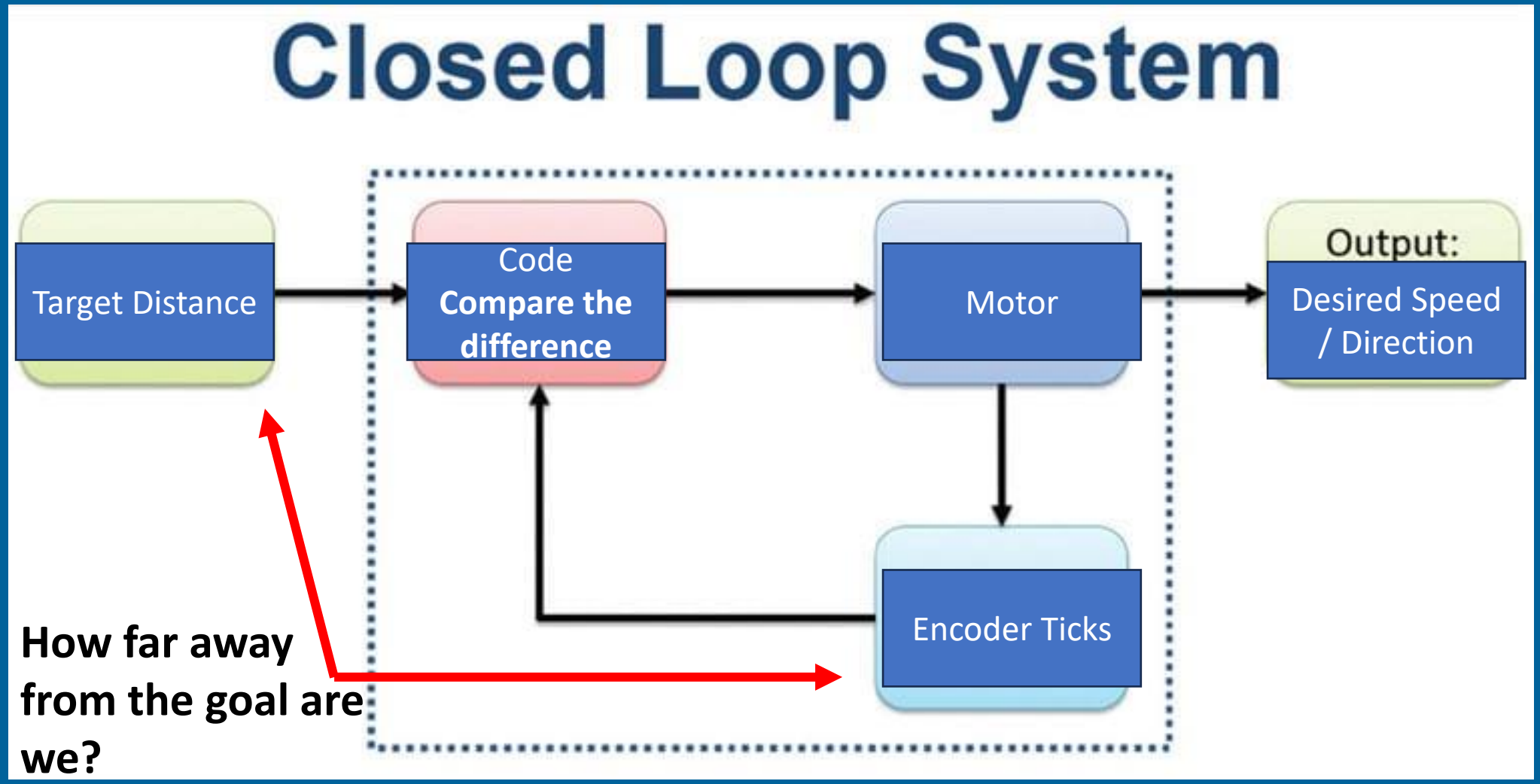
"Go until its at 1000 encoder counts!"
Not all motors are created equally

- Slow motor?

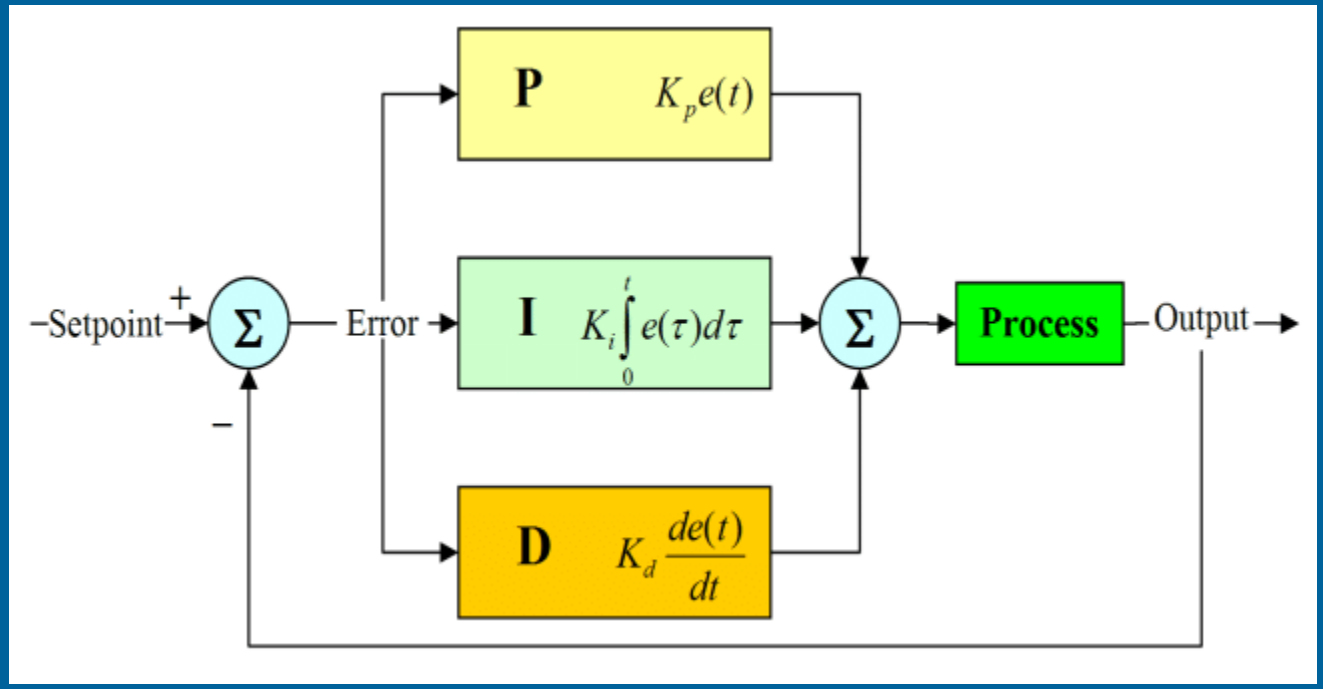
PID 1 – The Control Loop



PID 1 – The Control Loop



PID 2 – Setting our target

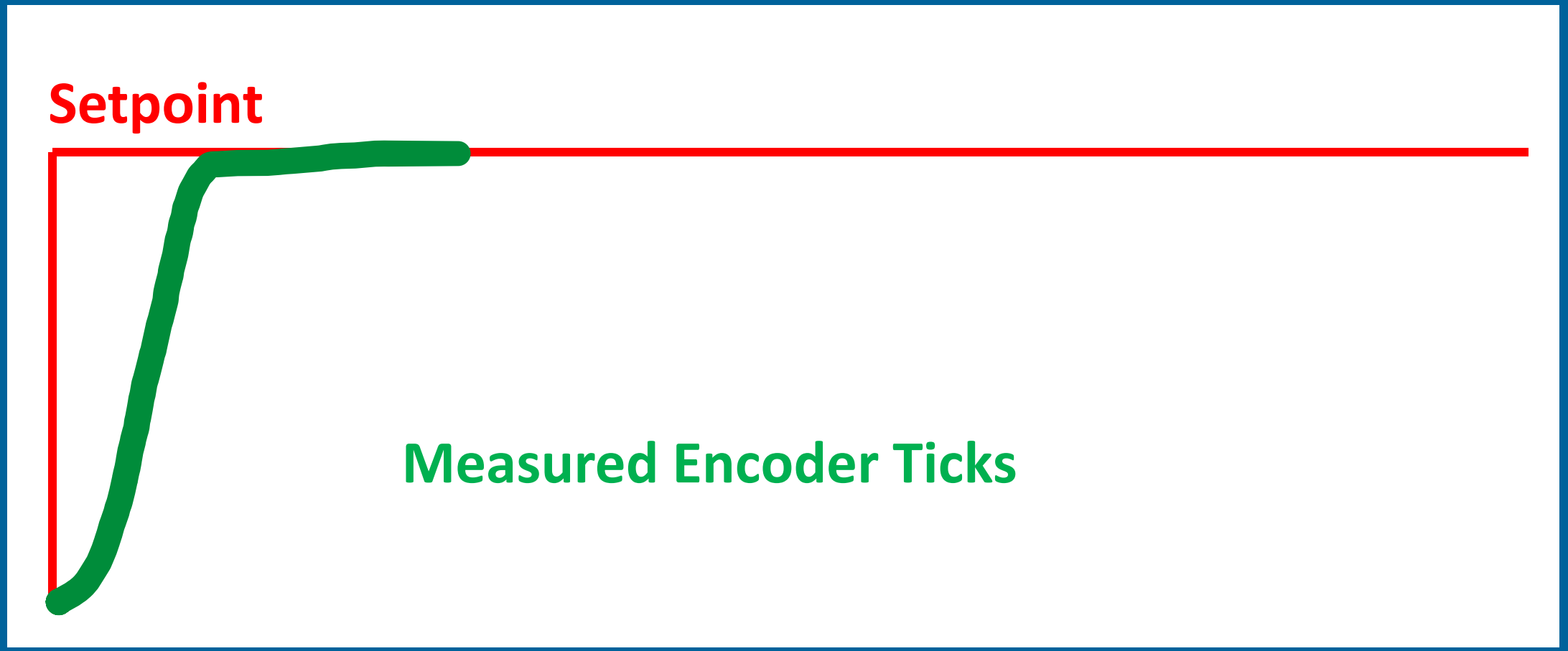


$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau)\,\mathrm{d}\tau + K_\mathrm{d} \frac{\mathrm{d}e(t)}{\mathrm{d}t},$$

Error = Setpoint – Current Encoder Ticks

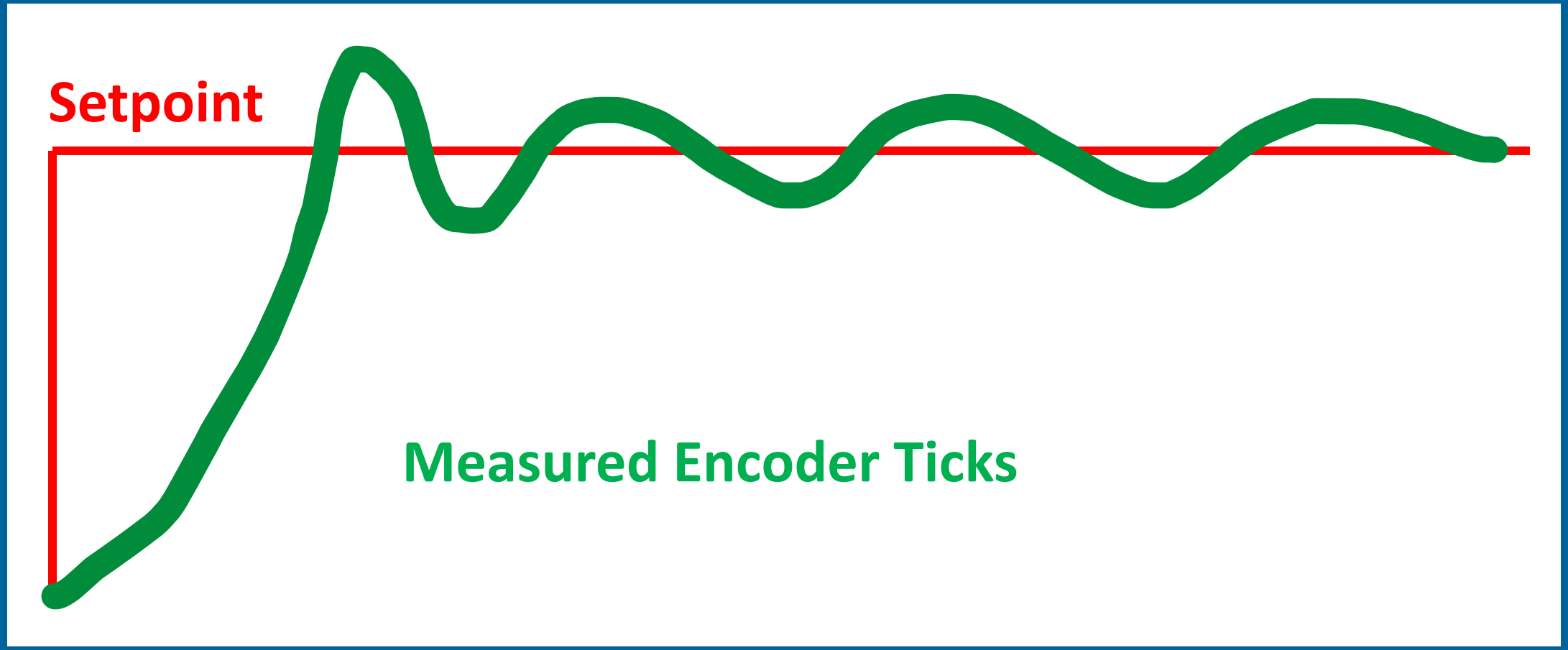
IEEE

PID 3: Proportion

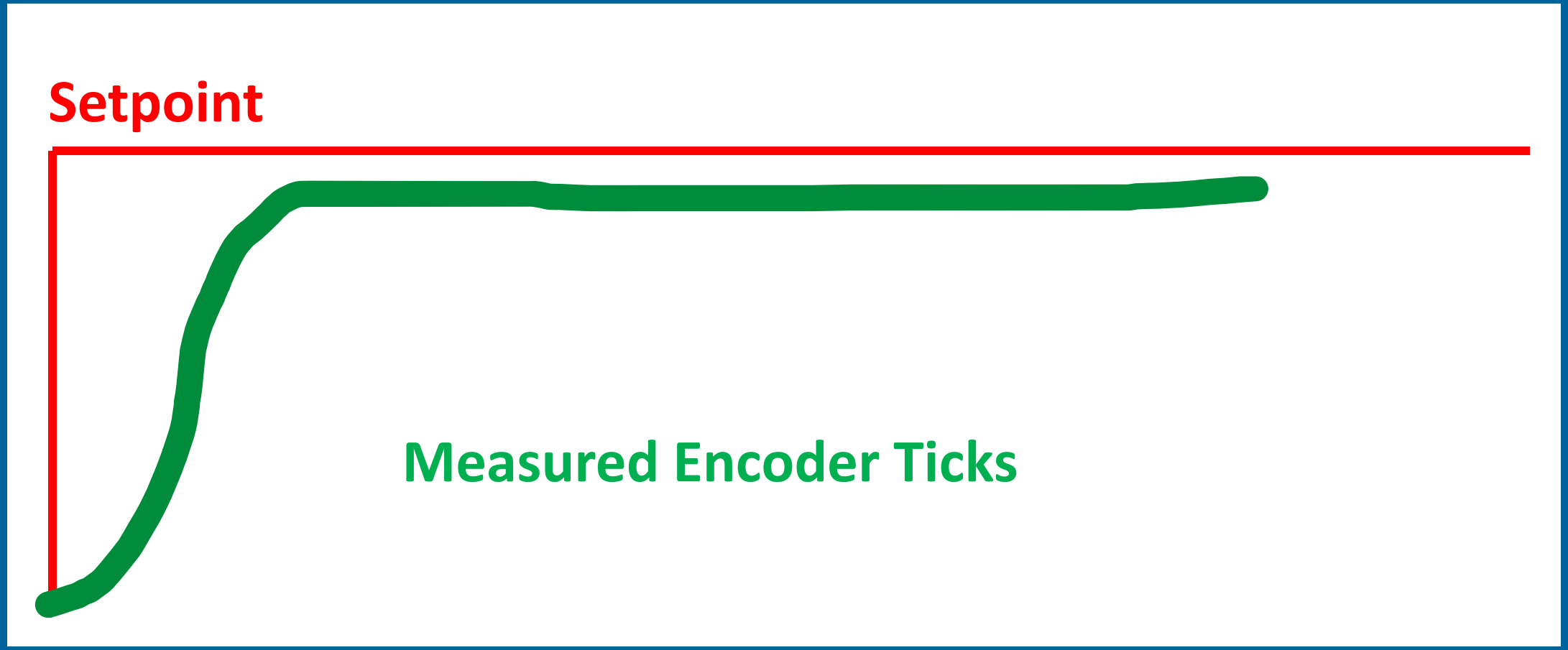


Error = Setpoint – Current Encoder Ticks

PID 4 – Dealing with overshoot & oscillation



PID 5 – Steady State Error



PID 6 – Coding a PID function

```
172  void motorPID(int setPoint, float kp, float ki, float kd){
173    int currentTime = micros();
174    int deltaT = ((float)(currentTime - prevTime)) / 1.0e6; // time difference between ticks in seconds
175    prevTime = currentTime; // update prevTime each loop
176
177    int error = setPoint - rightEncoderPos;
178    int errorDerivative = (error - prevError) / deltaT;
179    errorIntegral = errorIntegral + error*deltaT;
180
181    float u = kp*error + ki*errorIntegral + kd*errorDerivative;
182
183    float speed = fabs(u);
184    if(speed > 255){
185      speed = 255;
186    }
187
188    int dir = 1;
189    if (u < 0) {
190      dir = -1; // Move backward
191    } else {
192      dir = 1; // Move forward
193    }
194
195    setMotors(dir, speed);
196    prevError = 0;
197  }
```

IEEE

Calculating a target distance in real units

Things we know:
Motor is 20:1 gear ratio
Wheels are 32mm diameter
Encoder is 6 pole = 3 ticks per rotation

Calculating a target distance in real units Homework Assignment:

3 encoder ticks = 1 motor rotation
Wheel Rotations = motor rotation x gear ratio
Distance (mm) = wheel circumference x wheel rotations

Create a function in Arduino that converts encoder ticks to distance in mm or cm.

STARTER CODE ON GITHUB:
https://github.com/ieeecity/micromouse2024/