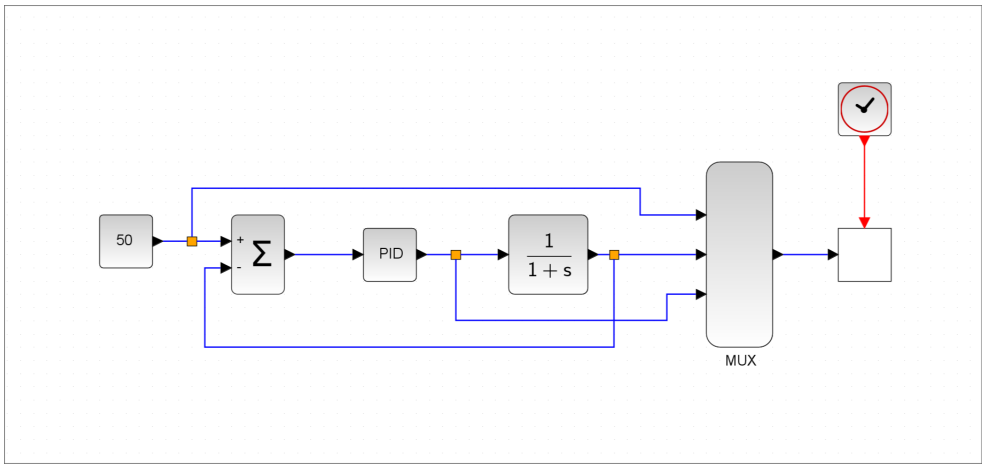
Digital version

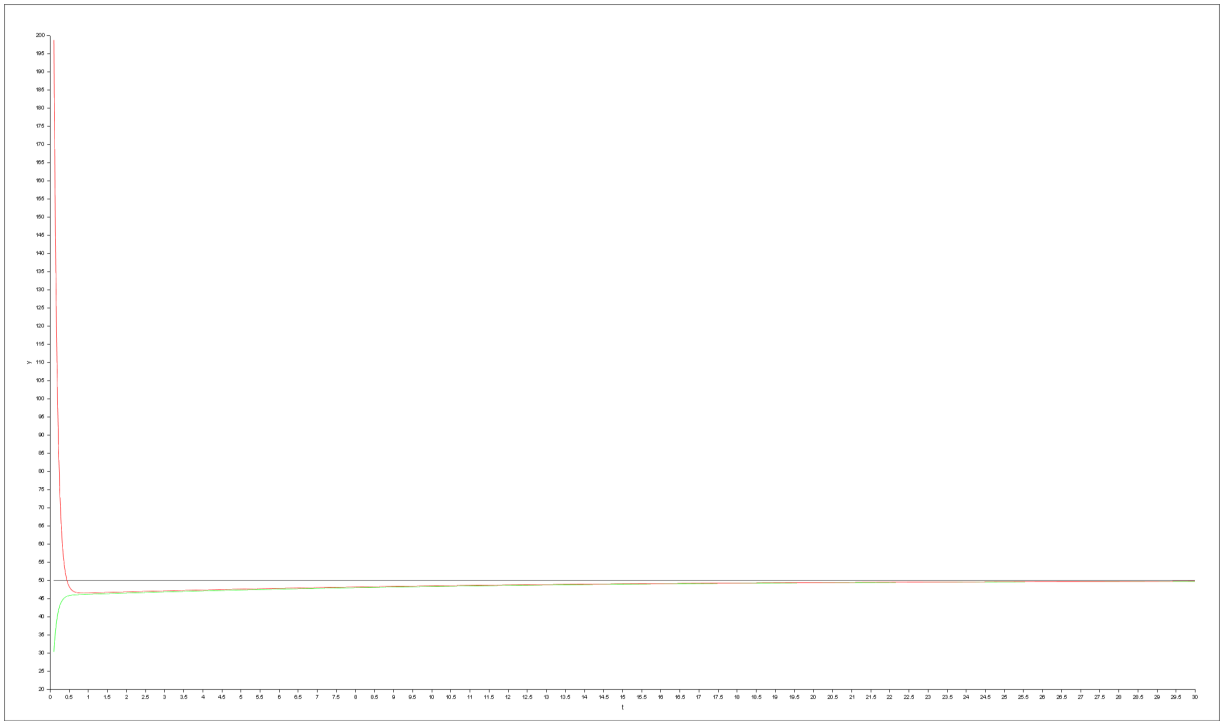
ENGS262 Control Systems Lab - Lab 2

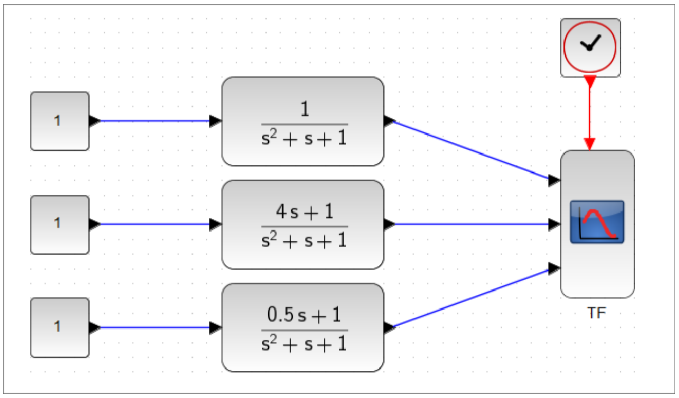
Mher Saribekyan Mane Hambardzumyan

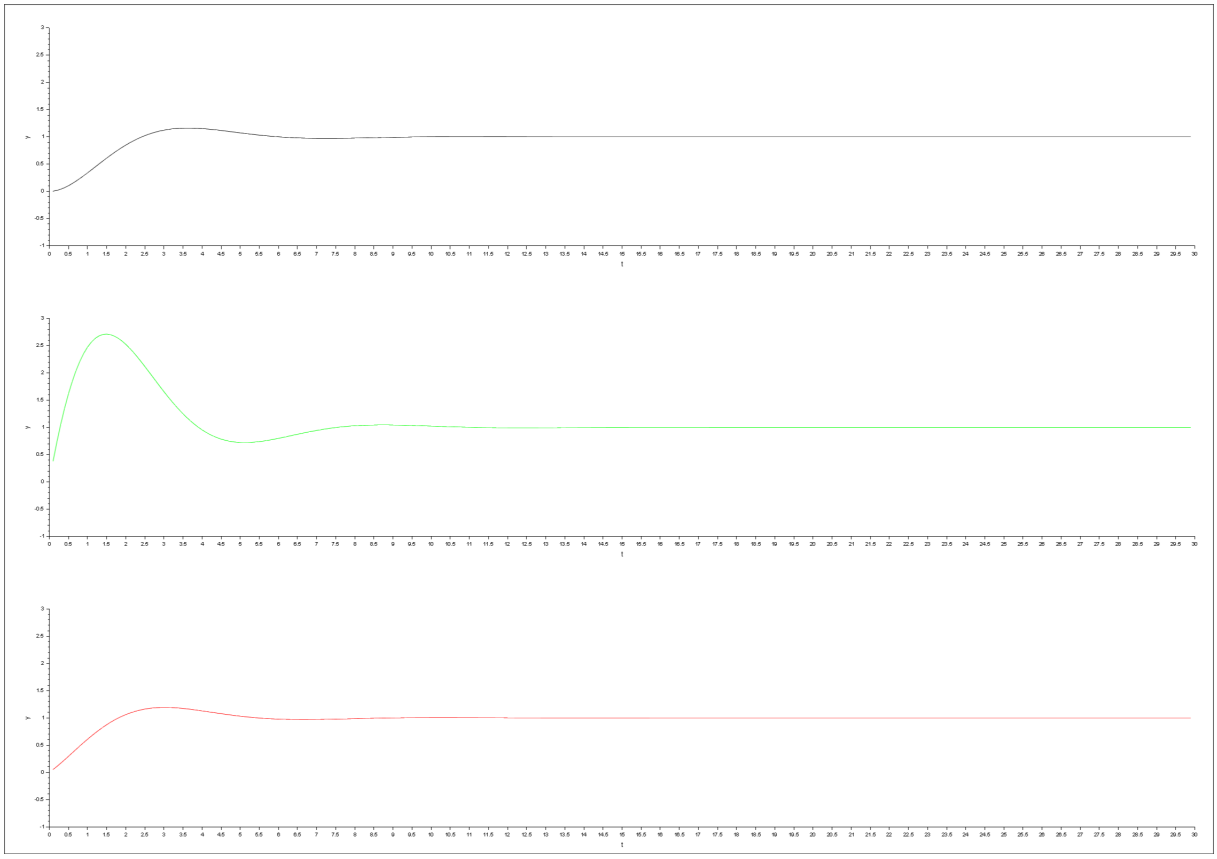
October 23, 2026

Lab 5









All the transfer functions have two imaginary poles, $-\frac{1}{2} \pm \frac{\sqrt{3}}{2}$. The first function has no zero, the second one has one zero at $s = -0.25$, and the thid one has a zero at $s = -2$. Since the second function has a zero close to the poles, it changes the transient and we get a slower response. The third function has a zero that is further from the poles and gives a faster response. The first function has no zero and gives the fastest response.

Lab 6

```
#include <ArduinoMotorCarrier.h>

int8_t direction = 1;
int8_t value = 0;

void setup()
{
  Serial.begin(115200);
  if (controller.begin())
  {
    Serial.print("OK");
  }
  else
  {
    Serial.print("Init error");
    while(1);
  }
  controller.reboot();
  delay(500);
}

void loop()
{
  if (value == 100)
  {
    direction = -1;
  }
  else if (value == -100)
  {
    direction = 1;
  }

  if (direction > 0)
  {
    value += 1;
  }
  else
  {
    value -= 1;
  }

  M1.setDuty(value);

  delay(10);
}
```

Listing 1: Motor Control

```
#include <ArduinoMotorCarrier.h>
#define A1 23
#define A2 21

float voltage = 0;
int value;

void setup()
{
  Serial.begin(115200);
  if (controller.begin())
  {
    Serial.print("OK");
  }
  else
  {
    Serial.print("Init error");
    while(1);
  }
  controller.reboot();
  delay(500);
}

void loop()
{
  controller.ping();
  voltage = battery.getConverted();

  value = voltage * (A1 + A2) / 2;
  if (value > 100)
    value /= 2;
  M1.setDuty(value);
  delay(10);
}
```

Listing 2: Battery Voltage

```ino
#include <ArduinoMotorCarrier.h>
#define BASE_SPEED 20

int8_t direction = 0;
int encoder_value = 0;

void setup()
{
  Serial.begin(115200);
  if (controller.begin())
  {
    Serial.print("OK");
  }
  else
  {
    Serial.print("Init error");
    while(1);
  }
  controller.reboot();

  encoder1.resetCounter(0);
  delay(500);
  while (direction == 0)
  {
    controller.ping();
    encoder_value = encoder1.getRawCount();
    Serial.println(encoder_value);
    if (encoder_value > 0)
    {
      direction = 1;
      break;
    }
    if (encoder_value < 0)
    {
      direction = -1;
      break;
    }
  }
  M1.setDuty(direction * BASE_SPEED);
}

void loop()
{
  controller.ping();
  encoder_value = encoder1.getRawCount();
  Serial.println(encoder_value);
  delay(10);
}
```

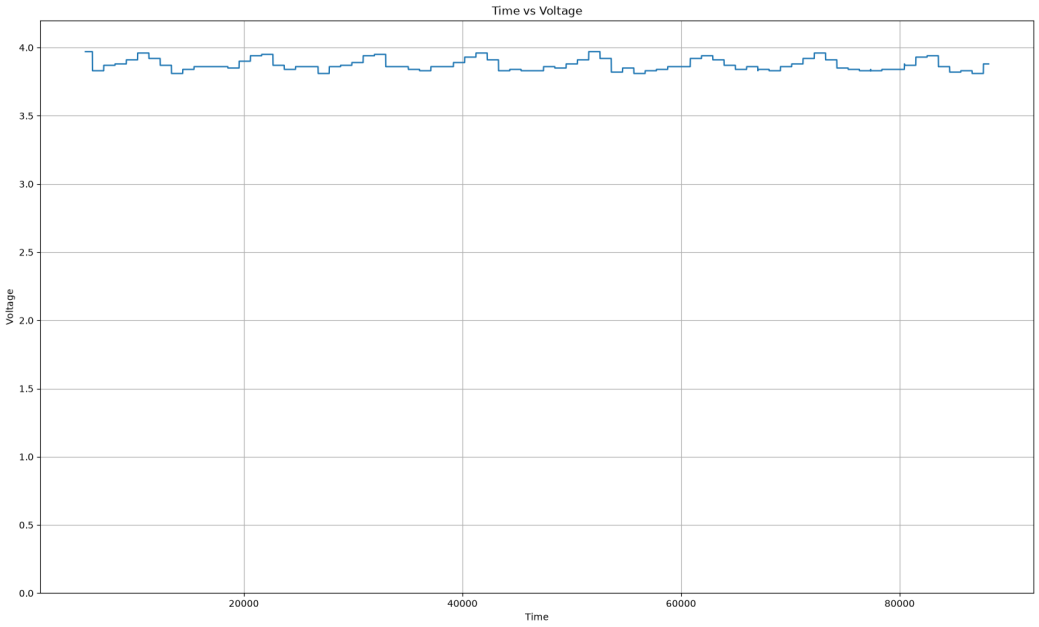
Listing 3: Encoder Test

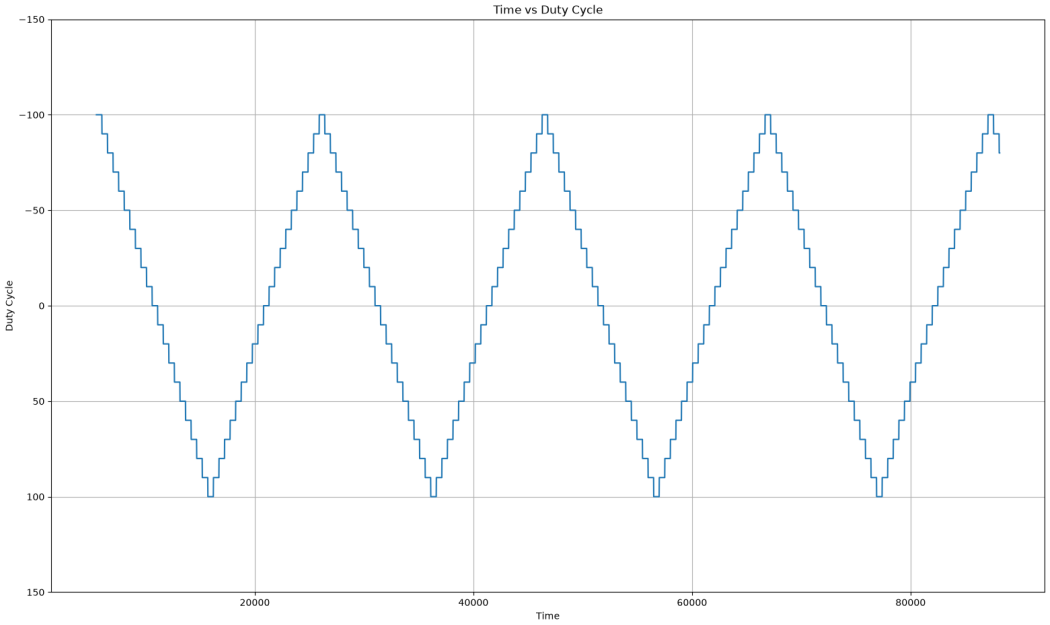
Lab 7

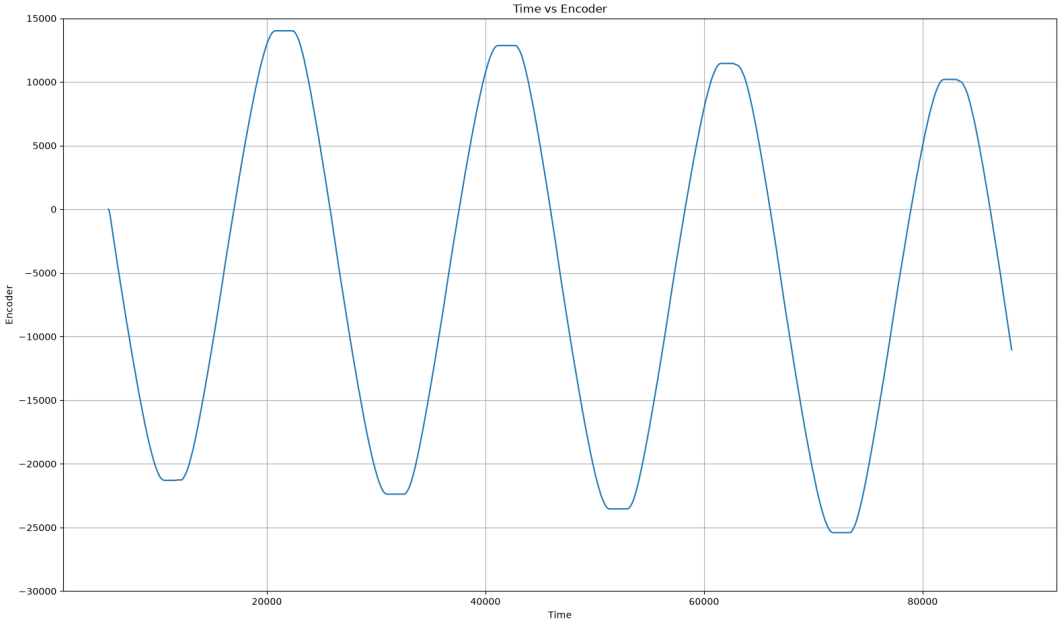
```
#include <ArduinoMotorCarrier.h>

int8_t direction = 1;
int8_t motor_speed = -100;
int encoder_value = 0;
int time_last_change = 0;

void setup()
{
  Serial.begin(115200);
  delay(5000);
  if (controller.begin())
  {
    Serial.print("OK");
  }
  else
  {
    Serial.print("Init error");
    while(1);
  }
  controller.reboot();

  encoder1.resetCounter(0);

  M1.setDuty(motor_speed);
  time_last_change = millis();
  Serial.println("time,voltage,duty_cycle,encoder_value");
}

void loop()
{
  if (time_last_change + 500 < millis())
  {
    if (motor_speed >= 100)
      direction = -1;
    else if (motor_speed <= -100)
      direction = 1;

    if (direction > 0)
      motor_speed += 10;
    else
      motor_speed -= 10;
    time_last_change = millis();
    M1.setDuty(motor_speed);
  }
  controller.ping();
  encoder_value = encoder1.getRawCount();
  Serial.print(millis());
  Serial.print(",");
  Serial.print((float) battery.getRaw()/236.0);
  Serial.print(",");
```

```
  Serial.print(motor_speed);
  Serial.print(",");
  Serial.println(encoder_value);
  delay(10);
}
```

Listing 4: Data Test



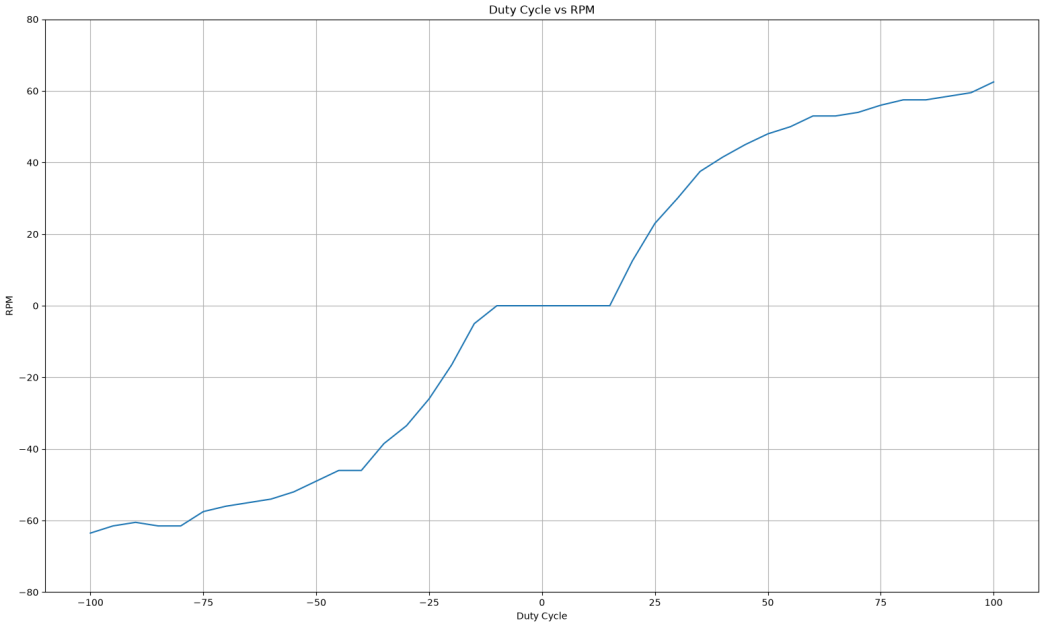

Time vs Duty Cycle
Duty Cycle
−150
−100
−50
0
50
100
150
20000
40000
60000
80000
Time


Time vs Encoder
Encoder
15000
10000
5000
0
−5000
−10000
−15000
−20000
−25000
−30000
20000
40000
60000
80000
Time

Lab 8

```
#include <ArduinoMotorCarrier.h>

// int8_t direction = 1;
int8_t motor_speed = -100;
int encoder_value = 0;
float rpm = 0;

void setup()
{
  Serial.begin(115200);
  delay(5000);
  if (controller.begin())
    Serial.print("OK");
  else
  {
    Serial.print("Init error");
    while(1);
  }
  controller.reboot();

  encoder1.resetCounter(0);

  M1.setDuty(motor_speed);
  Serial.println("time,duty_cycle,rpm");
  delay(1000);
}

void loop()
{
  controller.ping();
  encoder_value = encoder1.getRawCount();
  rpm = (float) (encoder1.getCountPerSecond() * 60)/ 48;
  Serial.print(millis());
  Serial.print(",");
  Serial.print(motor_speed);
  Serial.print(",");
  Serial.println(rpm);

  if (motor_speed >= 100)
  {
    M1.setDuty(0);
    while (1);
  }

  motor_speed += 5;
  M1.setDuty(motor_speed);

  delay(1000);
}
```

Listing 5: Linearity Test

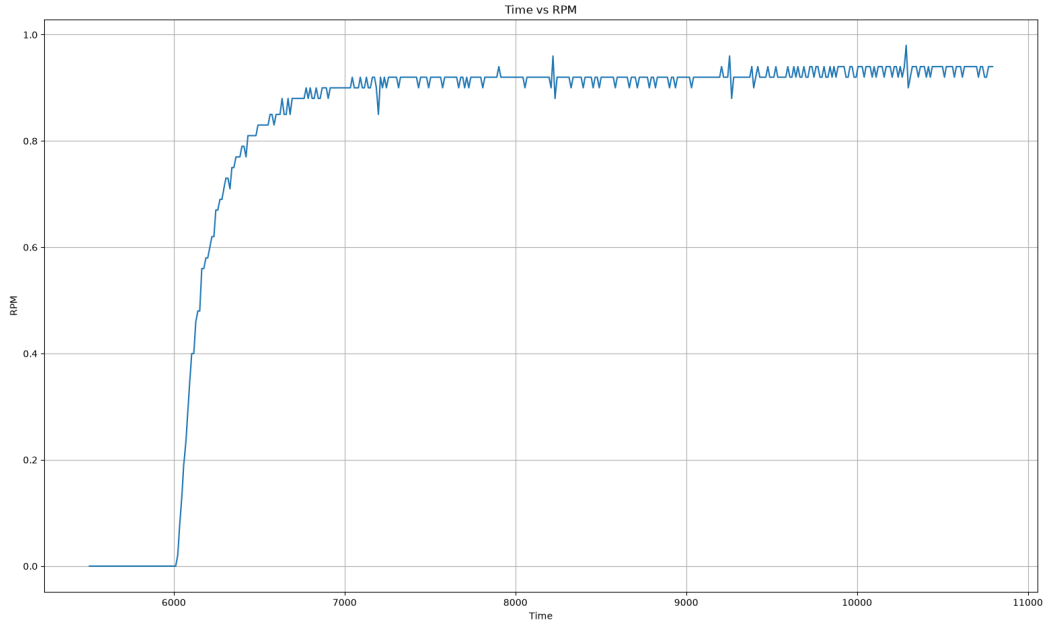


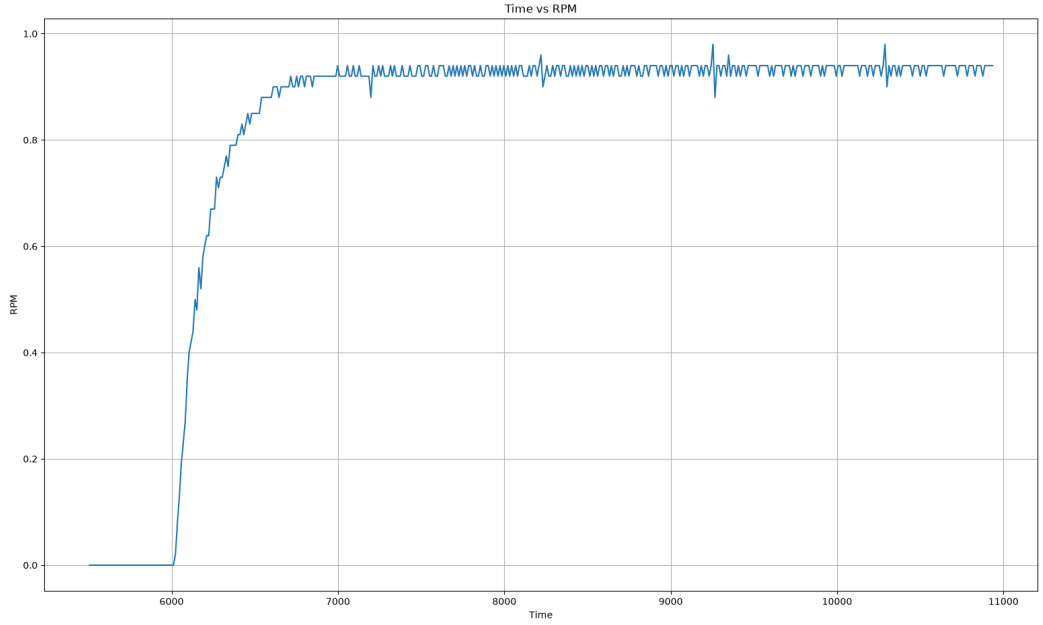
The system is not linear. We also have a deadband around the duty cycles of ± 20.

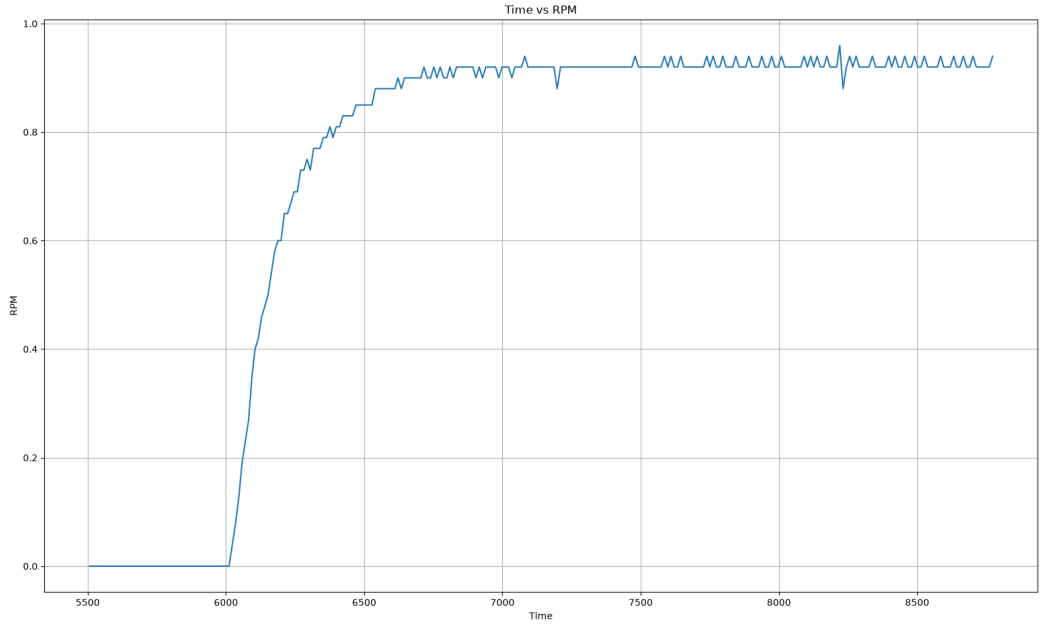
Lab 9

```
#include <ArduinoMotorCarrier.h>

bool started = 0;

void setup()
{
  Serial.begin(115200);
  delay(5000);
  if (controller.begin())
  {
    Serial.print("OK");
  }
  else
  {
    Serial.print("Init error");
    while(1);
  }
  controller.reboot();

  encoder1.resetCounter(0);

  Serial.println("time,rpm");
}

void loop()
{
  controller.ping();

  if (!started)
  {
    if (millis() > 6000)
    {
      M1.setDuty(50);
      started = true;
    }
  }
  Serial.print(millis());
  Serial.print(",");
  Serial.println((float) (encoder1.getCountPerSecond() * 60 / 50) / 48);

  delay(10);
}
```

Listing 6: Step response


Time vs RPM
RPM
Time
1.0
0.8
0.6
0.4
0.2
0.0
6000
7000
8000
9000
10000
11000


Time vs RPM
RPM
Time
1.0
0.8
0.6
0.4
0.2
0.0
6000
7000
8000
9000
10000
11000



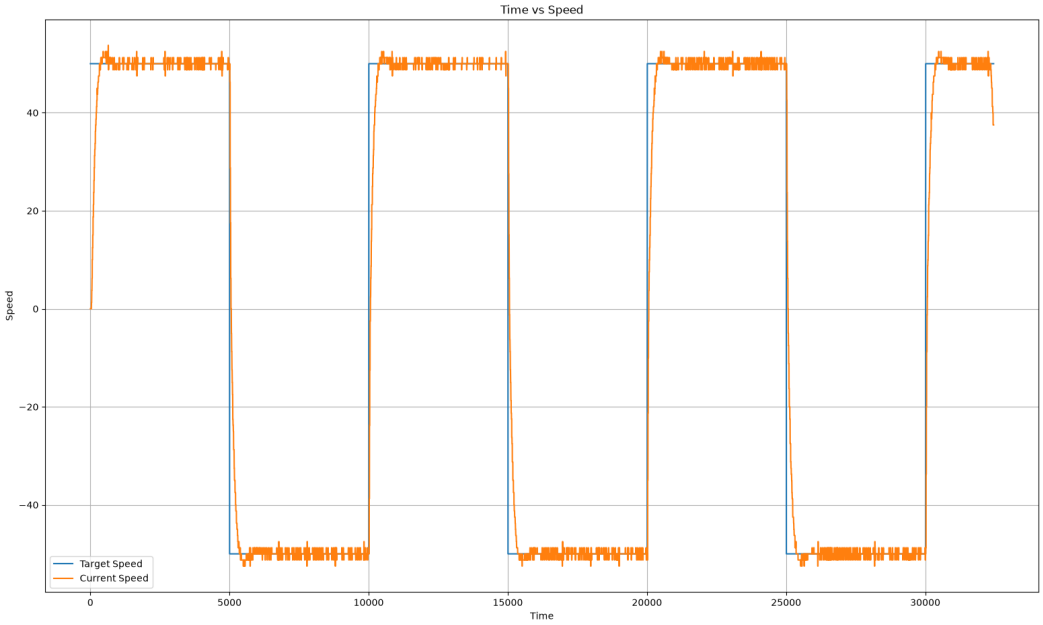
The motor starts spinning 6 seconds after power on. We get the following time constants, by finding the 63.2% points: $\tau_1 = 0.19s$, $\tau_2 = 0.19s$ and $\tau_3 = 0.18s$. The gain K is the steady state RPM (or in this case RPM/50), which are $K_1 = 0.92$, $K_2 = 0.93$ and $K_3 = 0.92$. The values are almost identical.

Lab 10

```
#include <ArduinoMotorCarrier.h>

#define time_step_ms 5

float KP = 2;
float KI = 0.01;

float integrator = 0;
float proportional = 0;
float control_signal = 0;

int last_flip = 0;
int target_rpm = 50;

int last_step = 0;
int time_start = 0;

void setup()
{
  Serial.begin(115200);
  delay(5000);
  if (controller.begin())
  {
    Serial.println("OK");
  }
  else
  {
    Serial.println("Init error");
    while(1);
  }
  controller.reboot();

  encoder1.resetCounter(0);

  Serial.println("time,target_rpm,rpm");

  last_flip = millis();
  last_step = millis();
  time_start = millis();
}

void loop()
{
  controller.ping();

  if (last_flip + 5000 < millis())
  {
    target_rpm = -target_rpm;
    last_flip = millis();
  }

```

```
  while (last_step + time_step_ms < millis());
  last_step = millis();

  float current_rpm = (float) (encoder1.getCountPerSecond() * 60) / 48;
  float error = target_rpm - current_rpm;

  integrator += error * time_step_ms;
  control_signal = (KP * error) + (KI * integrator);

  M1.setDuty((int) control_signal);

  Serial.print(millis() - time_start);
  Serial.print(",");
  Serial.print(target_rpm);
  Serial.print(",");
  Serial.println(current_rpm);
}
```

Listing 7: Speed Control

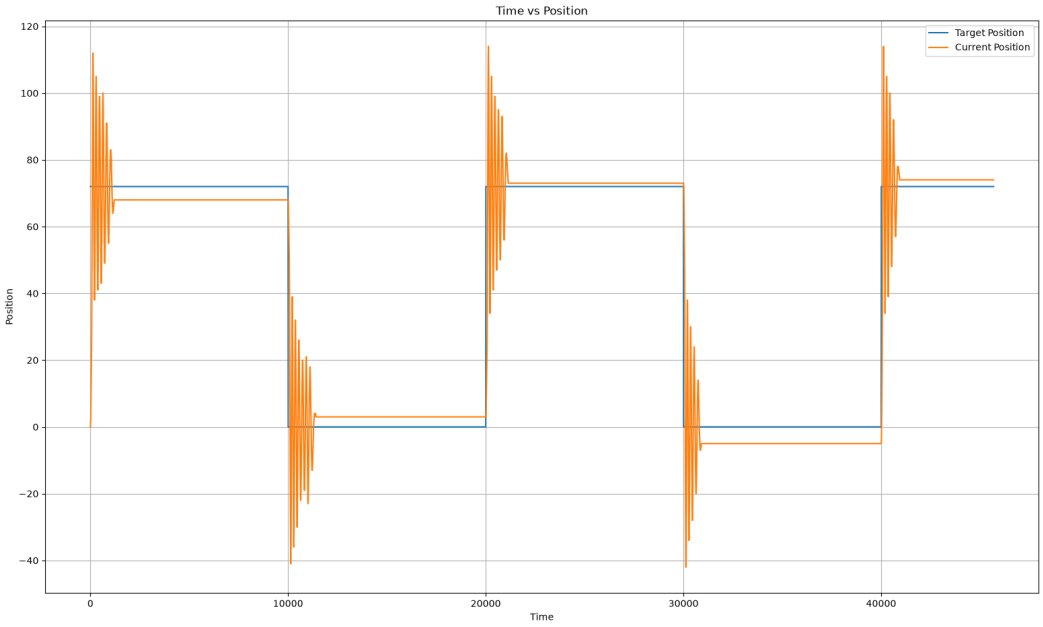


Lab 11

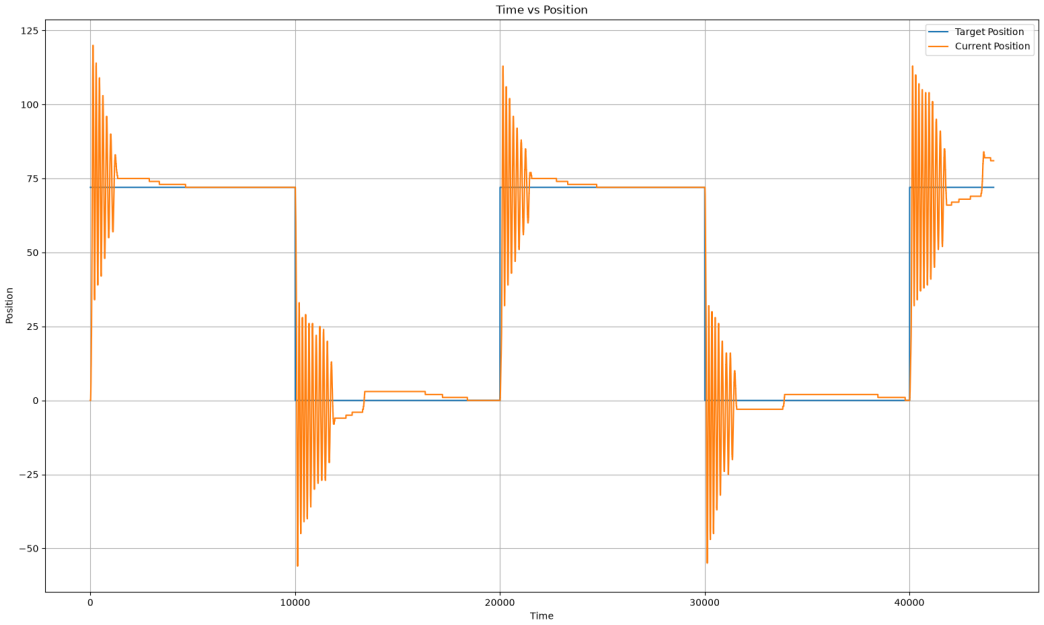
```
#include <ArduinoMotorCarrier.h>

#define time_step_ms 5

float KP = 2;
float KI = 0.001;

float integrator = 0;
float proportional = 0;
float control_signal = 0;

int last_flip = 0;
int target_count = 72;

int last_step = 0;
int time_start = 0;

void setup()
{
  Serial.begin(115200);
  delay(5000);
  if (controller.begin())
  {
    Serial.println("OK");
  }
  else
  {
    Serial.println("Init error");
    while(1);
  }
  controller.reboot();

  encoder1.resetCounter(0);

  Serial.println("time,target_count,count");

  last_flip = millis();
  last_step = millis();
  time_start = millis();
}

void loop()
{
  controller.ping();

  if (last_flip + 10000 < millis())
  {
    if (target_count == 0)
      target_count = 72;
    else
      target_count = 0;
```

```
    last_flip = millis();
  }

  while (last_step + time_step_ms < millis());
  last_step = millis();

  float current_count = encoder1.getRawCount();
  float error = target_count - current_count;

  integrator += error * time_step_ms;
  control_signal = (KP * error) + (KI * integrator);

  M1.setDuty((int) control_signal);

  Serial.print(millis() - time_start);
  Serial.print(",");
  Serial.print(target_count);
  Serial.print(",");
  Serial.println(current_count);
}
```

Listing 8: Position Control



With P controller only, the system does not reach the target value.



With PI controller, the system eventually reaches the target value.