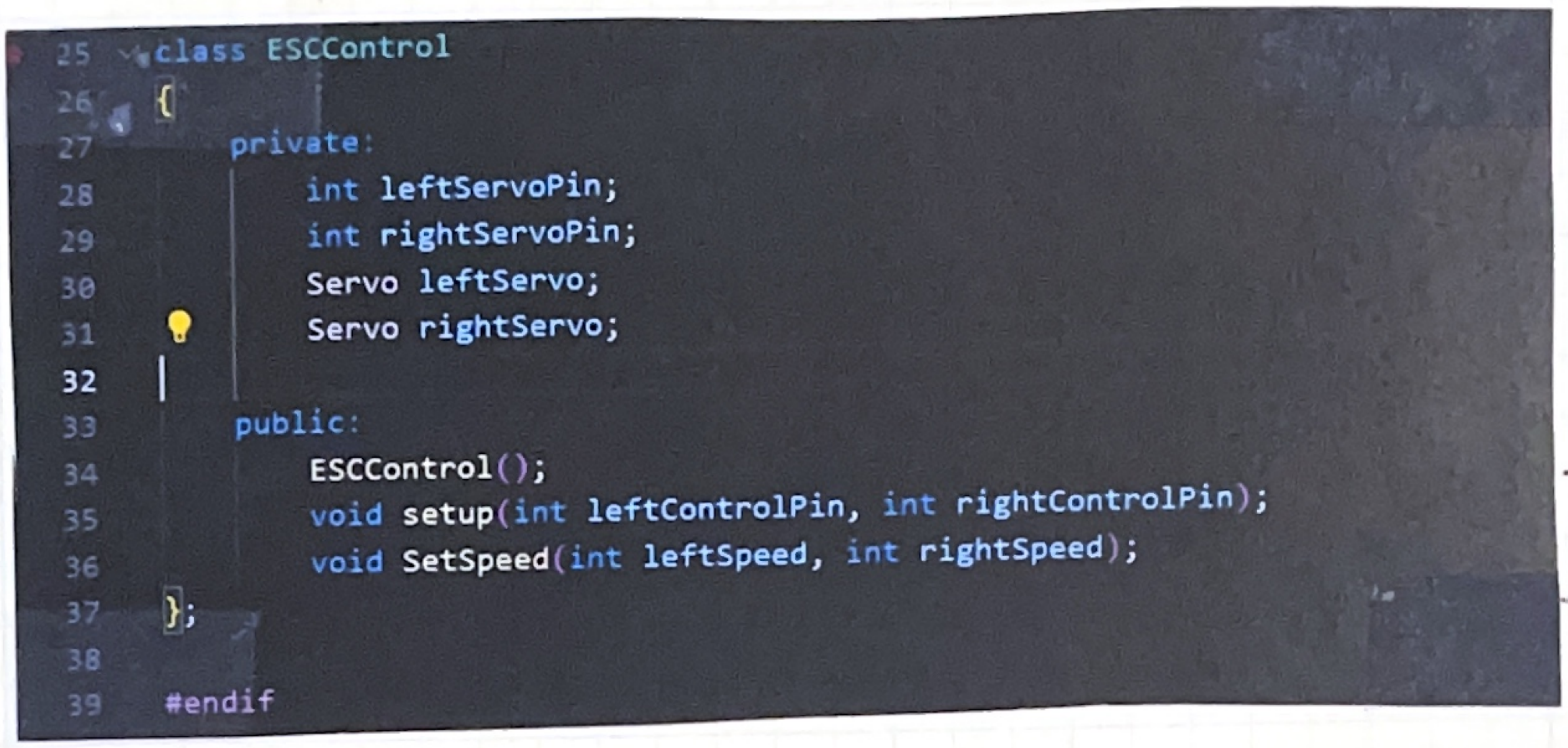
Dec 2023
Wesley Cook

Writing an ESC Library

We are going to extend the motor control H-Bridge library to include an option for Electronic Speed controllers. See page 43 for more details on the ESC.

First, let's modify "MotorControl.h" to tell arduino what we are planning to implement.



```
class ESCControl
{
    private:
        int leftServoPin;
        int rightServoPin;
        Servo leftServo;
        Servo rightServo;

    public:
        ESCControl();
        void setup(int leftControlPin, int rightControlPin);
        void SetSpeed(int leftSpeed, int rightSpeed);
};

#endif
```

} *

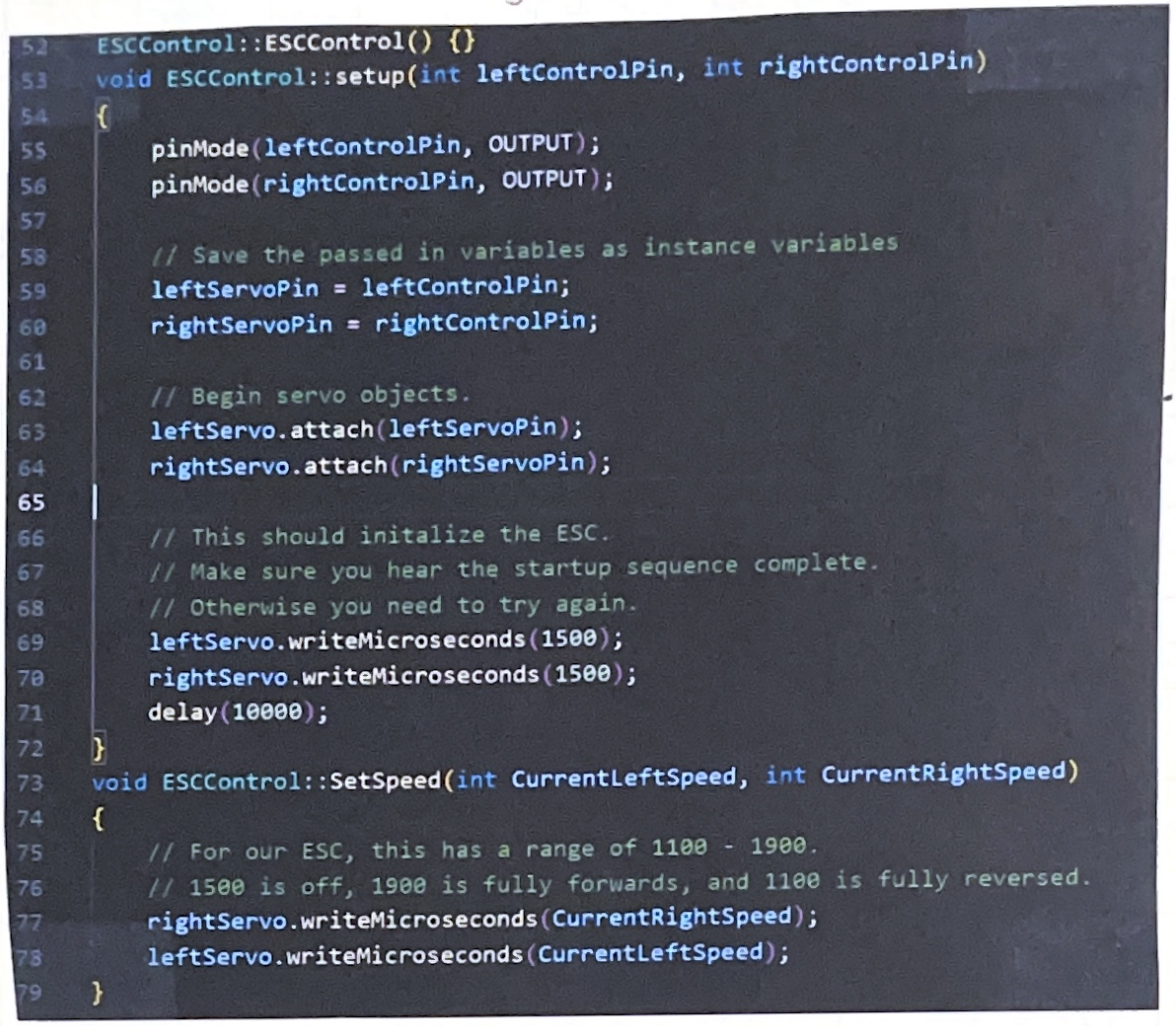
Here we are making a class called "ESC Control." The private variables are only accessable from inside the class definition. ~~Pu~~ Public variables and methods can be called from an instance of the class.

Here, we are going to have an empty constructor, and two public methods: "Setup" and "set speed"

"Setup" and "SetSpeed" are examples of prototypes. They give an outline of a method without actually implementing it.

Writing an ESC Library

Next, let's modify "MotorControl.cpp"



```
ESCControl::ESCControl() {}
void ESCControl::setup(int leftControlPin, int rightControlPin)
{
    pinMode(leftControlPin, OUTPUT);
    pinMode(rightControlPin, OUTPUT);

    // Save the passed in variables as instance variables
    leftServoPin = leftControlPin;
    rightServoPin = rightControlPin;

    // Begin servo objects.
    leftServo.attach(leftServoPin);
    rightServo.attach(rightServoPin);

    // This should initalize the ESC.
    // Make sure you hear the startup sequence complete.
    // Otherwise you need to try again.
    leftServo.writeMicroseconds(1500);
    rightServo.writeMicroseconds(1500);
    delay(10000);
}
void ESCControl::SetSpeed(int CurrentLeftSpeed, int CurrentRightSpeed)
{
    // For our ESC, this has a range of 1100 - 1900.
    // 1500 is off, 1900 is fully forwards, and 1100 is fully reversed.
    rightServo.writeMicroseconds(CurrentRightSpeed);
    leftServo.writeMicroseconds(CurrentLeftSpeed);
}
```

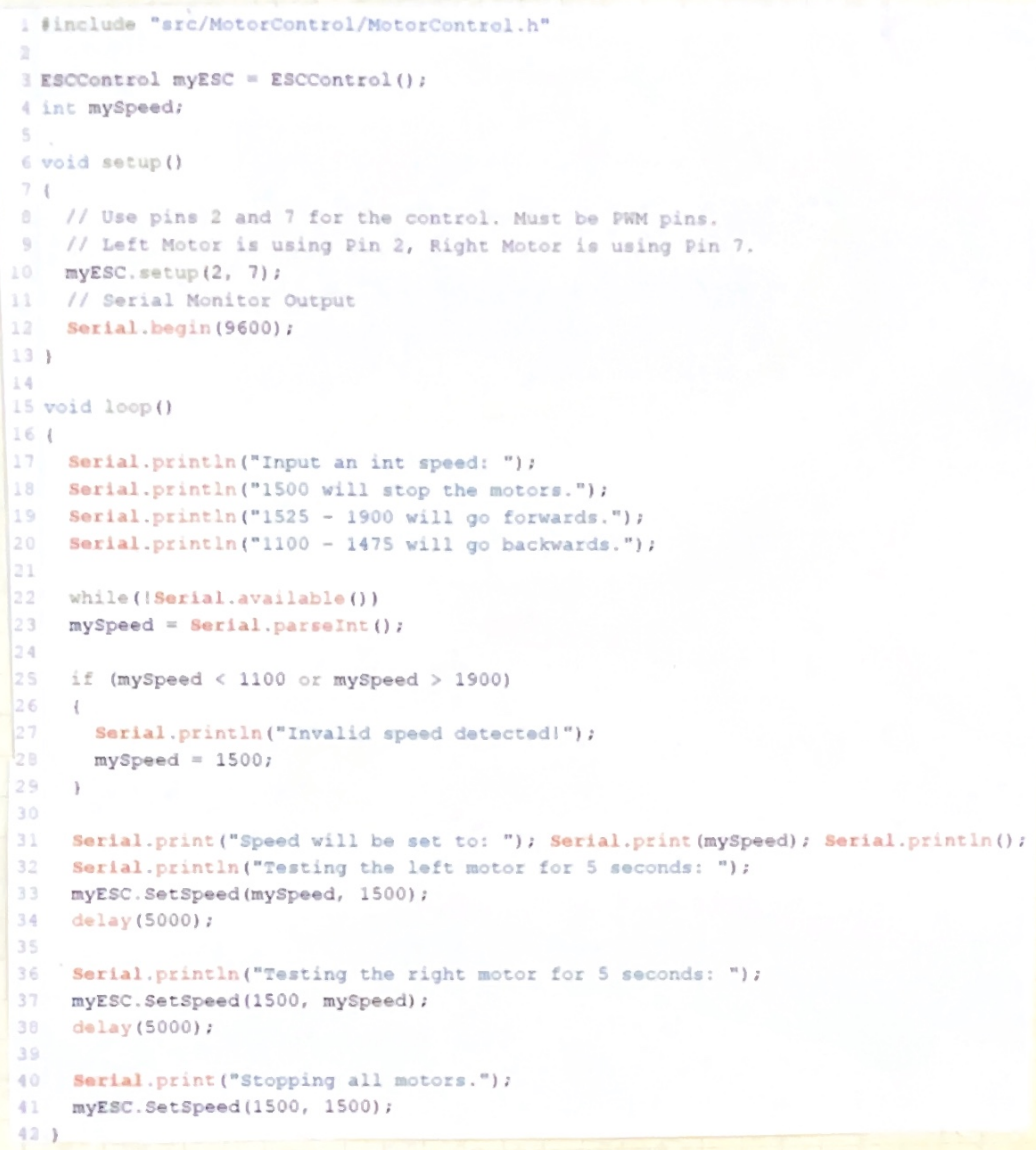
Here, we have an empty constructor so that we can choose when our ESC's should begin.

This is because of their special startup sequence

Dec 2023
Wesley
Cooke

Writing an ESC Library

Example Driver Code



```
#include "src/MotorControl/MotorControl.h"

ESCControl myESC = ESCControl();
int mySpeed;

void setup()
{
  // Use pins 2 and 7 for the control. Must be PWM pins.
  // Left Motor is using Pin 2, Right Motor is using Pin 7.
  myESC.setup(2, 7);
  // Serial Monitor Output
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Input an int speed: ");
  Serial.println("1500 will stop the motors.");
  Serial.println("1525 - 1900 will go forwards.");
  Serial.println("1100 - 1475 will go backwards.");

  while(!Serial.available())
  mySpeed = Serial.parseInt();

  if (mySpeed < 1100 or mySpeed > 1900)
  {
    Serial.println("Invalid speed detected!");
    mySpeed = 1500;
  }

  Serial.print("Speed will be set to: "); Serial.print(mySpeed); Serial.println();
  Serial.println("Testing the left motor for 5 seconds: ");
  myESC.SetSpeed(mySpeed, 1500);
  delay(5000);

  Serial.println("Testing the right motor for 5 seconds: ");
  myESC.SetSpeed(1500, mySpeed);
  delay(5000);

  Serial.print("Stopping all motors.");
  myESC.SetSpeed(1500, 1500);
}
```