# Motor Control Program - Robotics Course

## 1. Motor Control Variables

Make a new sketch.

The first thing to do is set which pins are used to control the motor driver. This is done by creating **variables**.

Variables have 3 elements:
1. Data Type (is it text or numbers)
2. A Name (should be easily understood)
3. Contents (what information is the variable representing)

For example: `int leftSpeedPin = 10;` , here we define a variable called `leftSpeedPin` which is an 'integer' ( `int` ) variable (i.e. a whole number) with a value of `10` which indicates the pin number for the Left Motor.

We need to make a new variable for each of the 6 control pins. We should store these above the `setup()` function.

```
int leftSpeedPin = 10;
int rightSpeedPin = 5;

int leftCwPin = 9;
int leftCcwPin = 8;
int RightCwPin = 7;
int RightCcwPin = 6;

void setup() {
```

## 2. Setting Pin Modes

Next, we need to tell the Arduino that we will be sending or **Outputting** signals from these pins. We do this using the `pinMode()` function.

This function takes two arguments, the pin number (we can use our variables) and whether it is an `INPUT` or an `OUTPUT` pin. Ours will be `OUTPUT` pins, but for sensors, we would use `INPUT` pins.

We should set all 6 pins as `OUTPUT` within the `setup()` function so it runs once when the Arduino is powered on.

```
void setup(){
    pinMode(leftSpeedPin, OUTPUT);
    pinMode(rightSpeedPin, OUTPUT);
    pinMode(leftCwPin, OUTPUT);
    pinMode(leftCcwPin, OUTPUT);
    pinMode(RightCwPin, OUTPUT);
    pinMode(RightCcwPin, OUTPUT);
}
```

## 3. Speed Commands

Next thing to do is start sending some instructions. First, let’s set the Both Motors to some high speed.

To do this, we will need to send 6 instructions, 2 Analogue (for speed) and 4 digital (for direction).

To send an analogue signal, we will use the `analogWrite()` function, this function takes two arguments, the pin number and the signal strength to send. The signal strength is a number between `0` and `255` where `0` is 0v and `255` is 5v, we can go anywhere between those to control the speed!

Now to make sure we don’t damage the batteries, we should limit the maximum speed to `220`. Let’s send that speed to both Motors. We should do this in the `loop()` function.

```
void loop(){
    analogWrite(leftSpeedPin, 220);
    analogWrite(rightSpeedPin, 220);
}
```

# 4. Direction Commands

Now we must also set the direction of each motor using the `digitalWrite()` function. This function takes two arguments, the pin number and whether it outputs a `HIGH` (5v) or `LOW` (0v) signal.

To set both motors clockwise:

```
void loop(){
    analogWrite(leftSpeedPin, 220);
    analogWrite(rightSpeedPin, 220);

    digitalWrite(leftCwPin, HIGH);
    digitalWrite(leftCcwPin, LOW);

    digitalWrite(RightCwPin, HIGH);
    digitalWrite(RightCcwPin, LOW);
}
```

Now, plug in a 9v battery to power the motor driver, upload the code to the Arduino and test it out!

# 5. Experimentation

Get cadets to vary the speeds on the `analogWrite()` functions and to vary the directions using the `digitalWrite()` functions.

Get them to think about what set of commands will make the motors turn, go backwards, stop etc.

# 6. Custom Functions

It can get cumbersome to write out all 6 commands every time we want to make a change, we can make our own functions to put at the end of our Arduino sketch.

Here is a function for driving clockwise:

```
void forwards(int speed){
    //Left Clockwise
    digitalWrite(leftCwPin, HIGH);
    digitalWrite(leftCcwPin, LOW);

    //Right Clockwise
    digitalWrite(RightCwPin, HIGH);
    digitalWrite(RightCcwPin, LOW);

    //Output Speed
    analogWrite(leftSpeedPin, speed);
    analogWrite(rightSpeedPin, speed);
}
```

`void` here indicates that the function does not output anything, it just runs.

`speed` here is an input that we use whenever we are calling the variable, notice that it is an `int` so it is a whole number. When we call the `forwards()` function, we also define a speed.

Here is an example:

```
void loop(){
    forwards(220);
}
```

Here we call the `forwards()` function with a speed of `220`, when this line is executed, it will jump to the function we made and execute each line, once it is done, it will go back to the `loop()` function.

# 7. Complex Control

**If cadets are struggling:** Direct to open the `Motor_Control.ino` file on the laptop desktop where functions are already written.

**Otherwise** Discuss what other functions we might want i.e.

```
forwards();
backwards();
left();
right();
stop();
```

Get cadets to write these functions and test them.

We can call these functions in the while loop to drive some path, let’s say we want to go forwards for 1 second, then turn left for half a second, we would write:

```
void loop(){
    forwards(220);
    delay(1000);

    left(220);
    delay(500);
}
```

This code will make the robot for forwards then left on repeat forever!

Now get cadets to edit their code to make some random path of their choosing.