American International University- Bangladesh

Department of Electrical and Electronic Engineering

EEE4103: Microprocessor and Embedded Systems Laboratory

Title: Controlling a motor through the application of PWM.

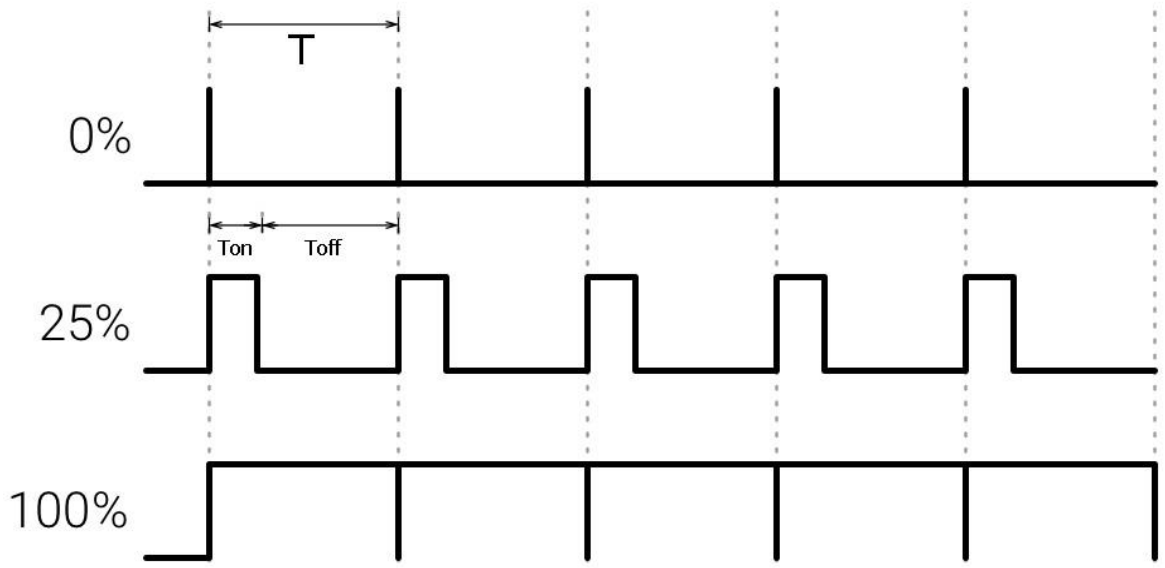
Introduction:

The objective of this experiment is to get familiarized with Microcontroller based motor speed control.

Theory and Methodology:

Microcontroller and Arduino are digital devices; they cannot give the analog output. Microcontroller gives Zero and ONE as output, where ZERO is logical LOW and ONE is logical HIGH. In our case, we are using 5 volt version of the Arduino. So it's logical ZERO is zero voltage, and logical HIGH is 5 voltage.

Digital output is good for digital devices but sometimes we need the analog output. In such a case the PWM is very useful. In the PWM, output signal switches between zero and one, on high and fixed frequency, as shown in the figure below.



Output Signal Of PWM

As shown in the above figure the ON time is T_{on} and the OFF time is T_{off}. **T is the sum of the T_{on} and T_{off}, which is called the Time Period.** In the concept of PWM, T is not varying and the T_{on} and the T_{off} can vary, in this way when T_{on} increase T_{off} will decrease and T_{off} increase when T_{on} decrease proportionally.

The duty cycle is the fraction of one Time period. Duty cycle is commonly expressed as a percentage or a ratio. A period is the time it takes for a signal to complete an on-and-off cycle. As a formula, a duty cycle may be expressed as:

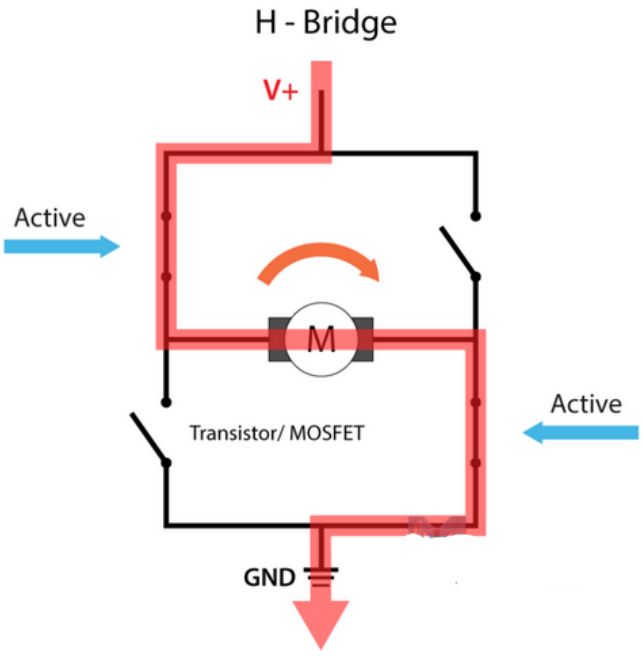
DUTY CYCLE = (Ton / T) x100 %

Now the motor speed varies according to duty cycle. Suppose the duty is zero, motor does not run and when duty cycle is 100 % the motor moves on maximum RPM**. But this concept is not always right because motor starts running after giving some fixed voltage that is called threshold voltage.**

Microcontroller and the Arduino can process signals and **consumes almost 20 to 40mA current but motors need high current and voltage, so we are using the transistor for driving the motor.** Transistor is connected in ***series*** with motor and ***transistor's base is connected to Arduino's PWM pin through a resistance.*** PWM signal is coming from Arduino and the transistor works as a switch and it **short circuits the Emitter (E) and Collector (C) when PWM signal is in High state** and ***normally opens when PWM signal is in LOW state***. This process works continuously and the motors runs at desired speed.

H-Bridge DC Motor Control

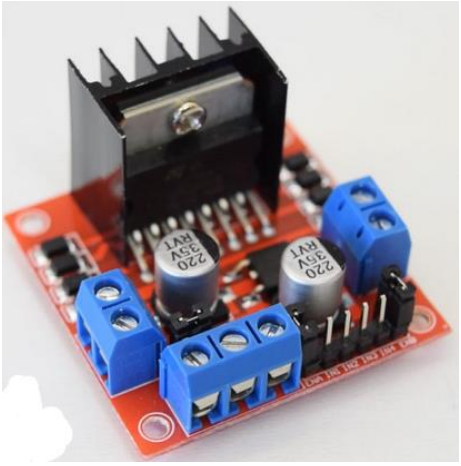
On the other hand, for controlling the rotation direction, we just need to inverse the direction of the current flow through the motor, and the most common method of doing that is by using an H-Bridge. An H-Bridge circuit contains four switching elements, transistors or MOSFETs, with the motor at the center forming an H-like configuration. **By activating two particular switches at the same time we can change the direction of the current flow, thus change the rotation direction of the motor.**



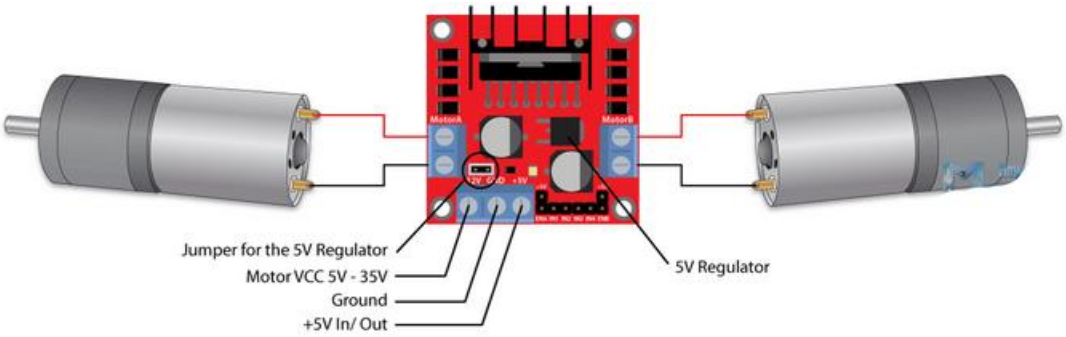
So if we combine these two methods, the PWM and the H-Bridge, we can have a complete control over the DC motor. There are many DC motor drivers that have these features and the L298N is one of them.

L298N Driver

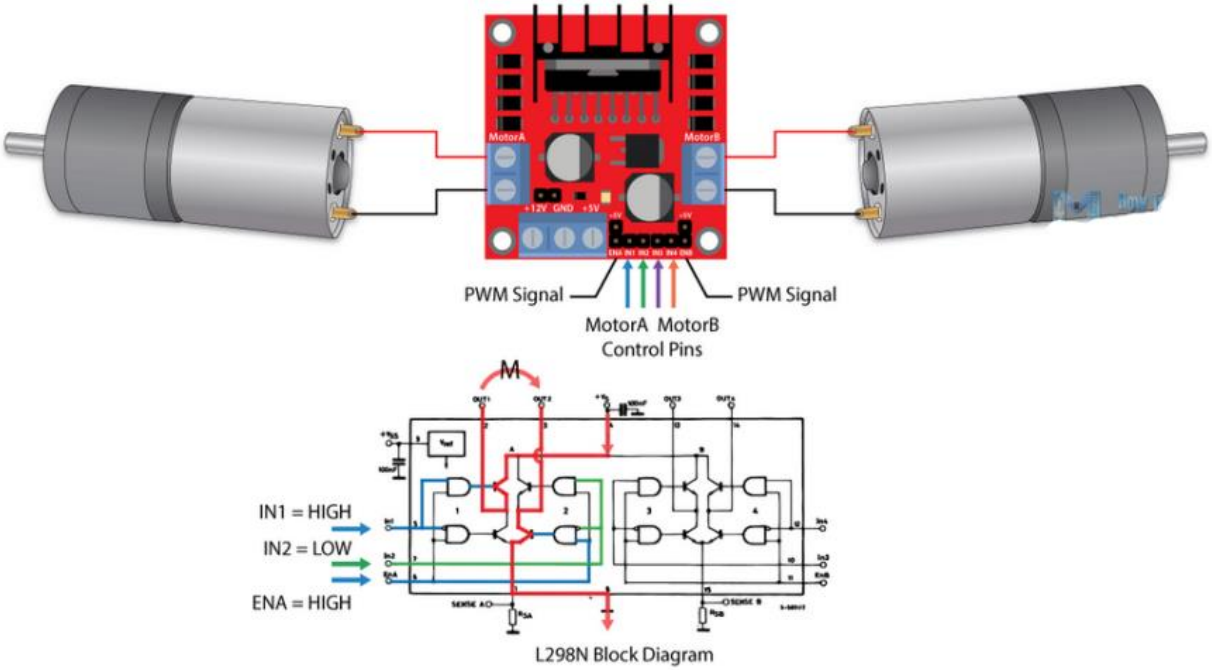
The L298N is a ***dual H-Bridge motor driver which allows speed and direction control of two DC motors at*** the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A.



Let's take a closer look at the pinout of L298N module and explain how it works. The module has two screw terminal blocks for the motor A and B, and another screw terminal block for the Ground pin, the VCC for motor and a 5V pin which can either be an input or output.



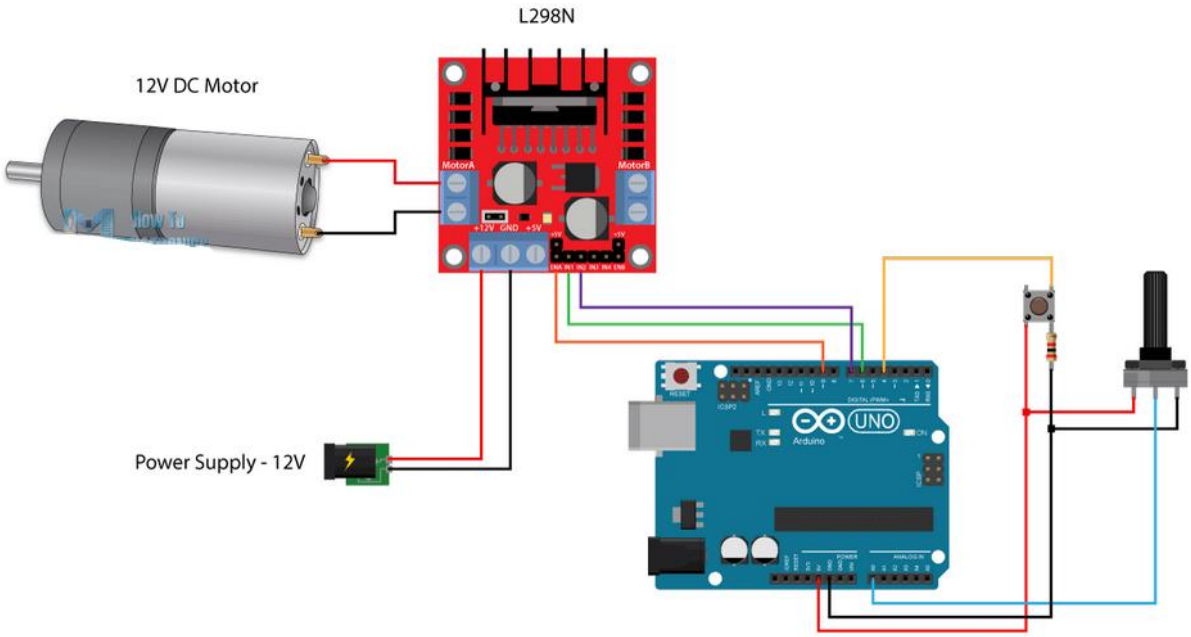
Next are the logic control inputs. The Enable A and Enable B pins are used for enabling and controlling the speed of the motor. ***If a jumper is present on this pin, the motor will be enabled and work at maximum speed, and if we remove the jumper we can connect a PWM input to this pin and in that way control the speed of the motor. If we connect this pin to a Ground the motor will be disabled.***



Next, the Input 1 and Input 2 pins are used for controlling the rotation direction of the motor A, and the inputs 3 and 4 for the motor B. Using these pins we actually control the switches of the H-Bridge inside the L298N IC. If input 1 is LOW and input 2 is HIGH the motor will move forward, and vice versa, if input 1 is HIGH and input 2 is LOW the motor will move backward. In case both inputs are same, either LOW or HIGH the motor will stop. The same applies for the inputs 3 and 4 and the motor B.

Arduino and L298N

Now let's make some practical applications. In the first example we will ***control the speed of the motor using a potentiometer*** and ***change the rotation direction using a push button***. Here's the circuit schematic diagram.



So we need an L298N driver, a DC motor, a potentiometer, a push button and an Arduino board.

Components List

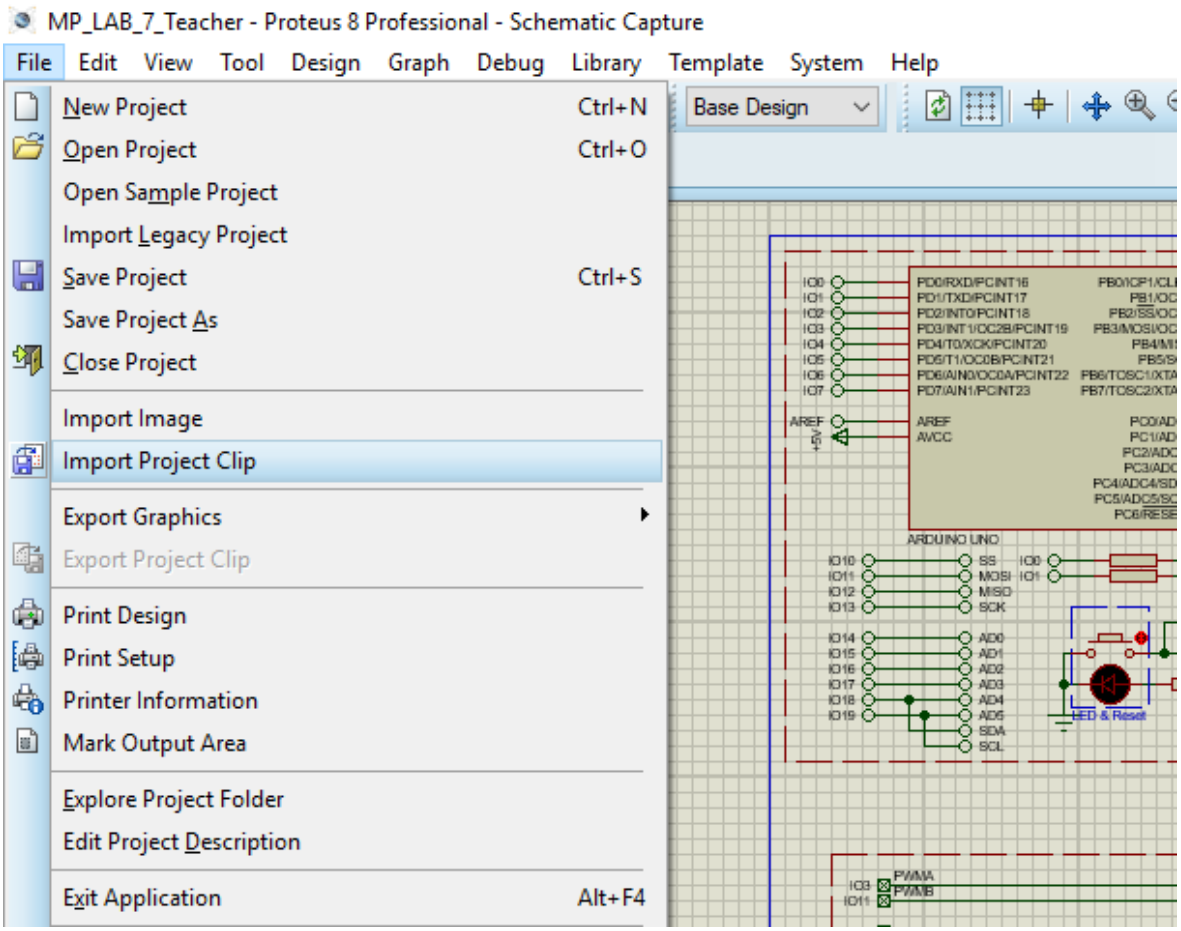
- L298N Driver
- 12V High Torque DC Motor
- Arduino Board
- Breadboard and Jump Wires

Experimental Procedure:

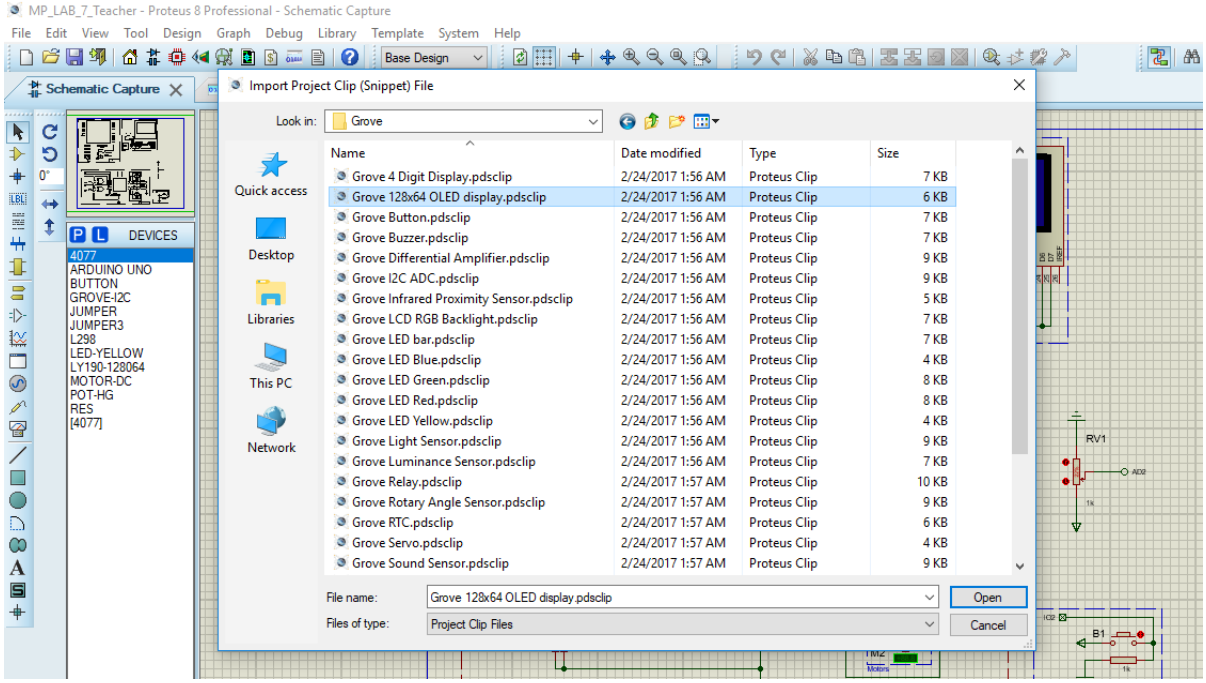
Simulation in Proteus:

Start by creating a new Visual Designer Project and then Add the Arduino Shield with DC Motors in the normal way.

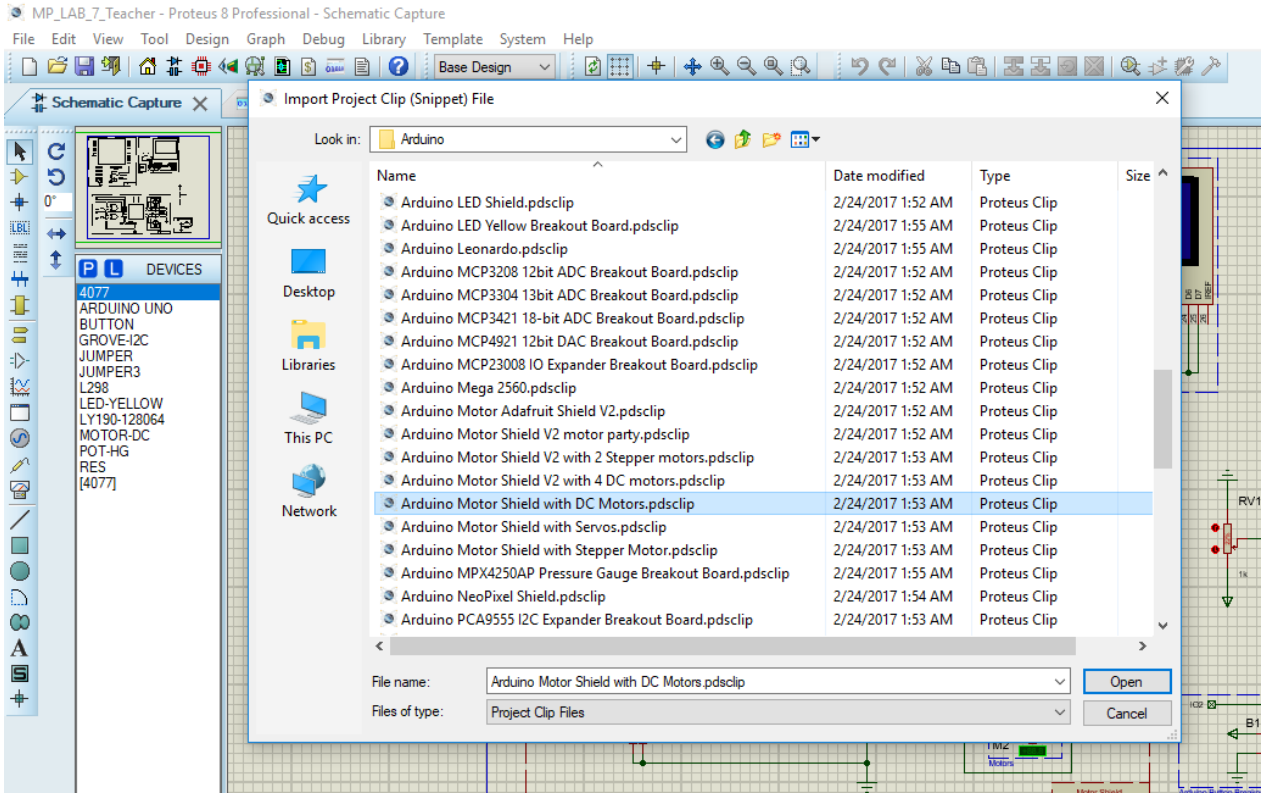
Go to File and select Import Project Clip.



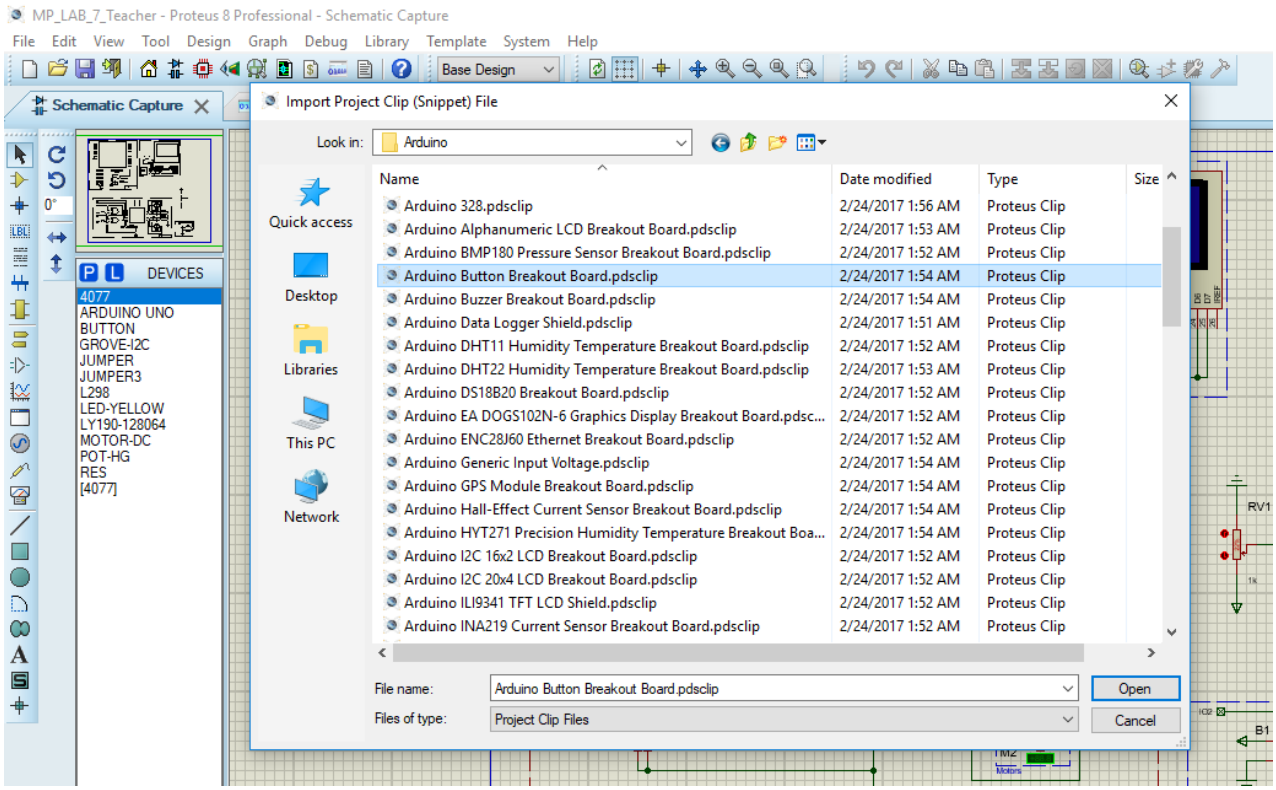
Then select Grove folder to select Grove 128×64 OLED display and place it in the schematic.



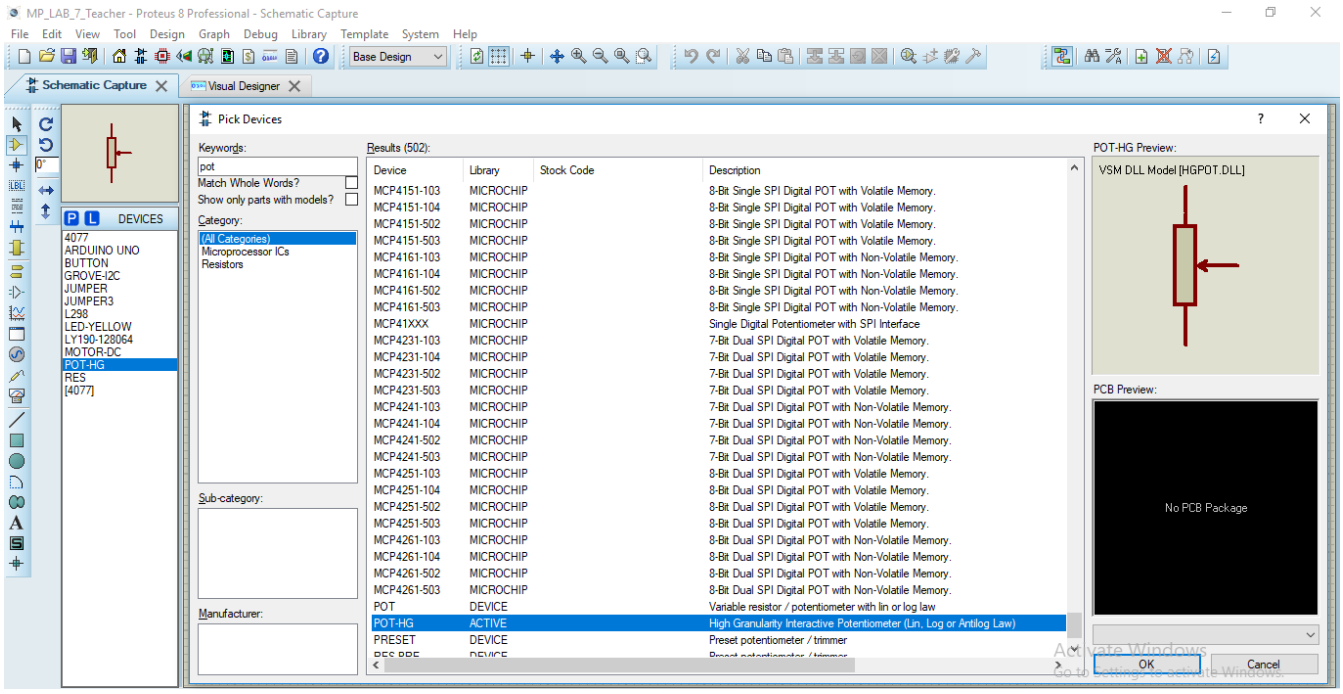
Then from File again select Arduino to select Arduino Motor Shield with DC Motors and then place it in the Schematic.



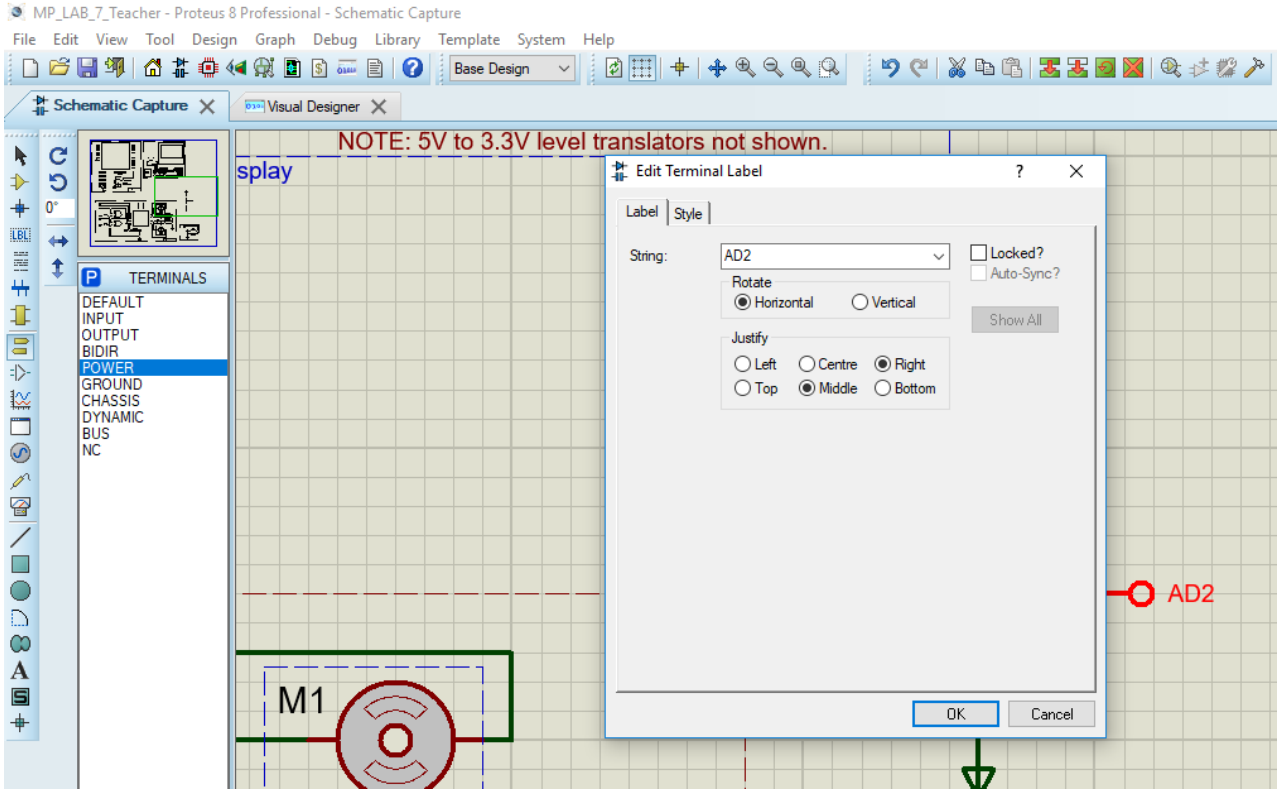
From Arduino again select Arduino Button Breakout Board and add that to schematic too.



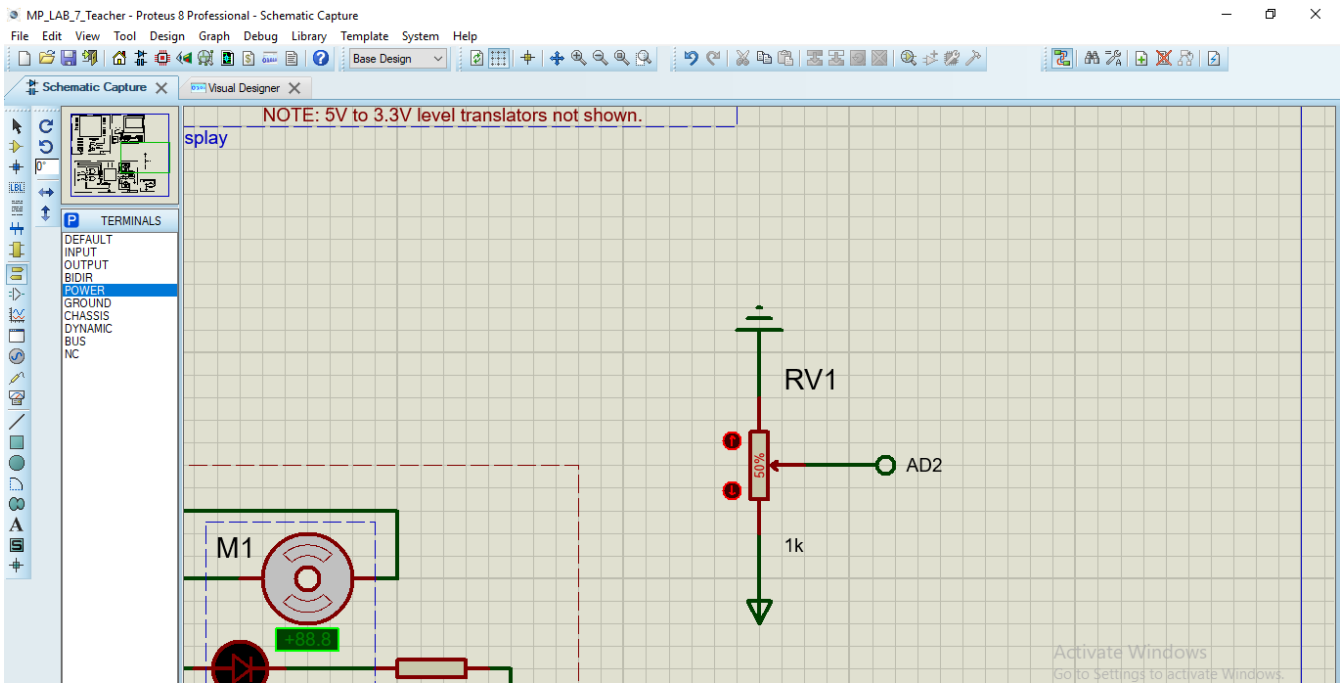
Turn on **component mode** on leftmost panel, select P (for picking devices). Write pot in keywords and select POT-HG and place into Schematics.



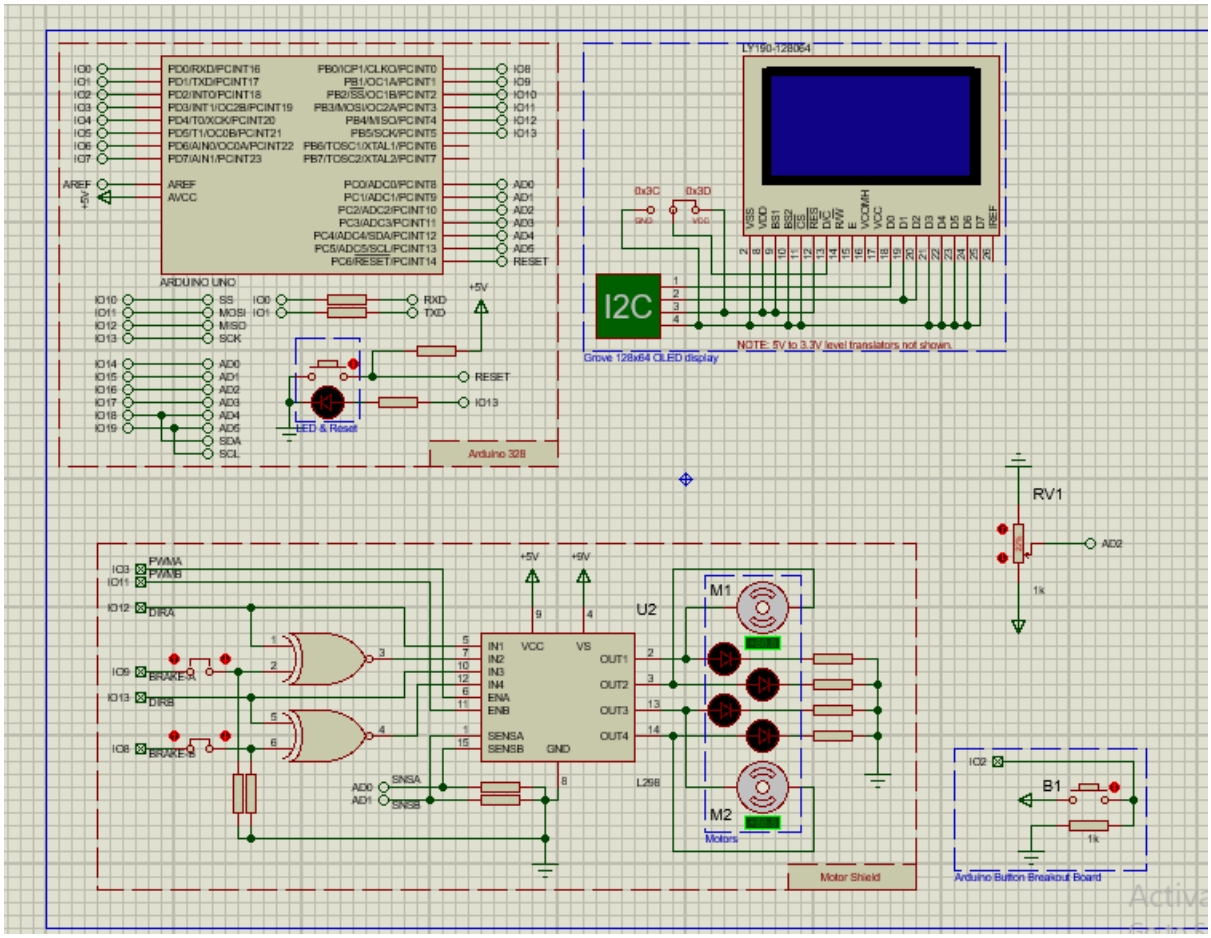
Then from terminal mode select ground and voltage supply and connect with POT-HG. Next select default to place a label and edit it to AD2.



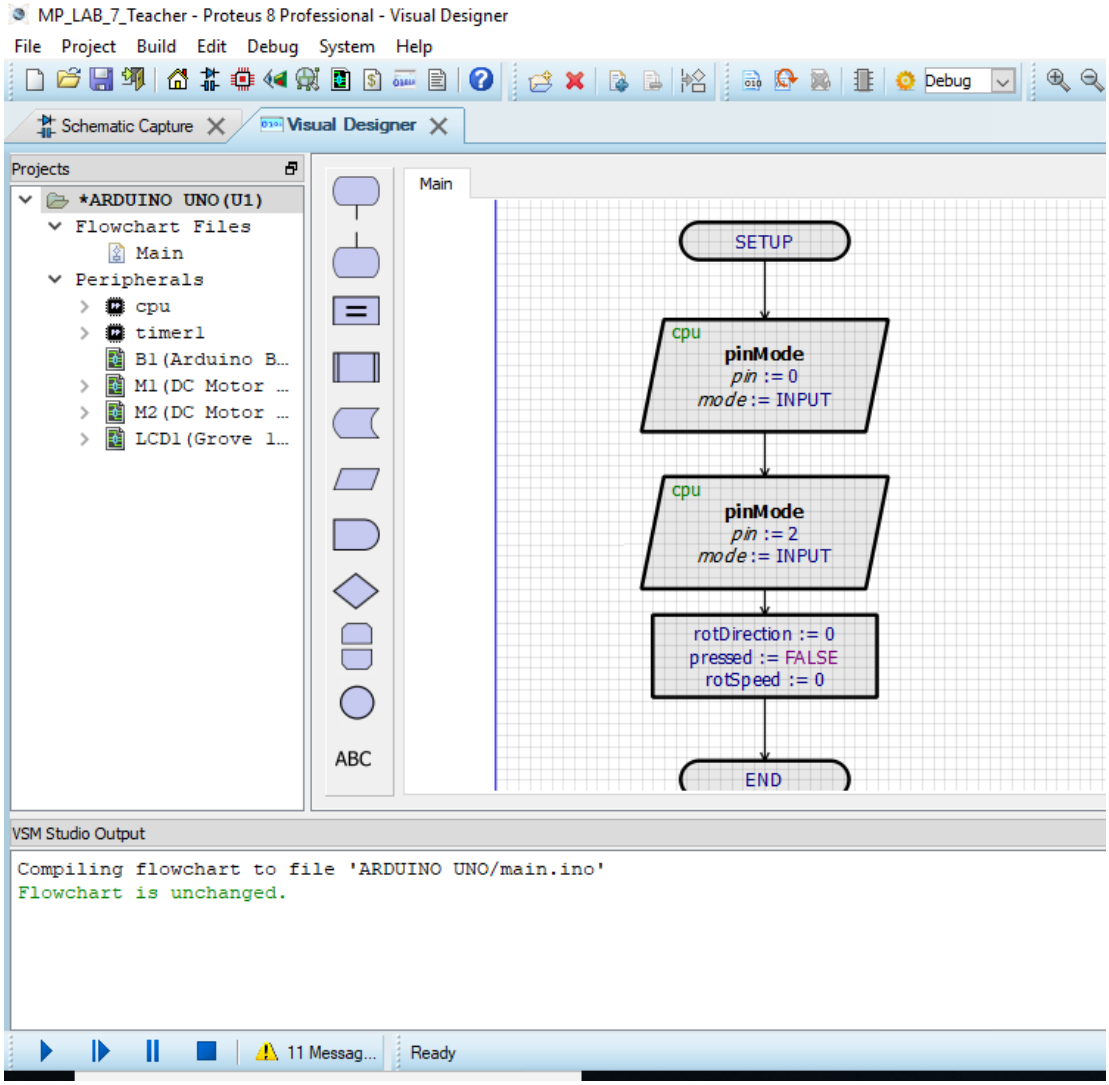
So the connection would look like below:



So The Full design will look like below:

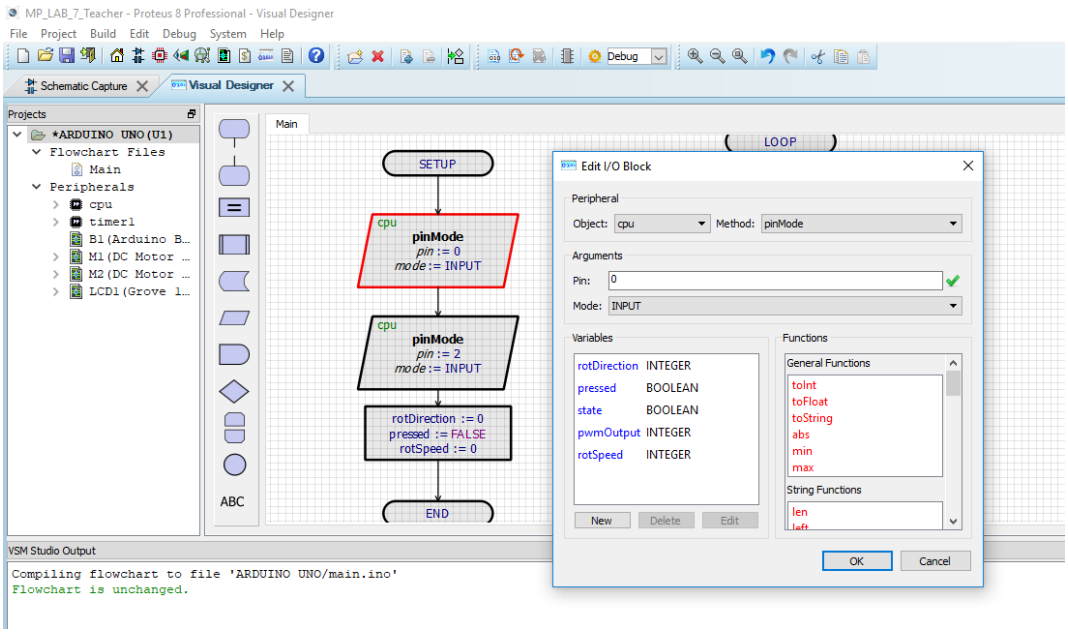


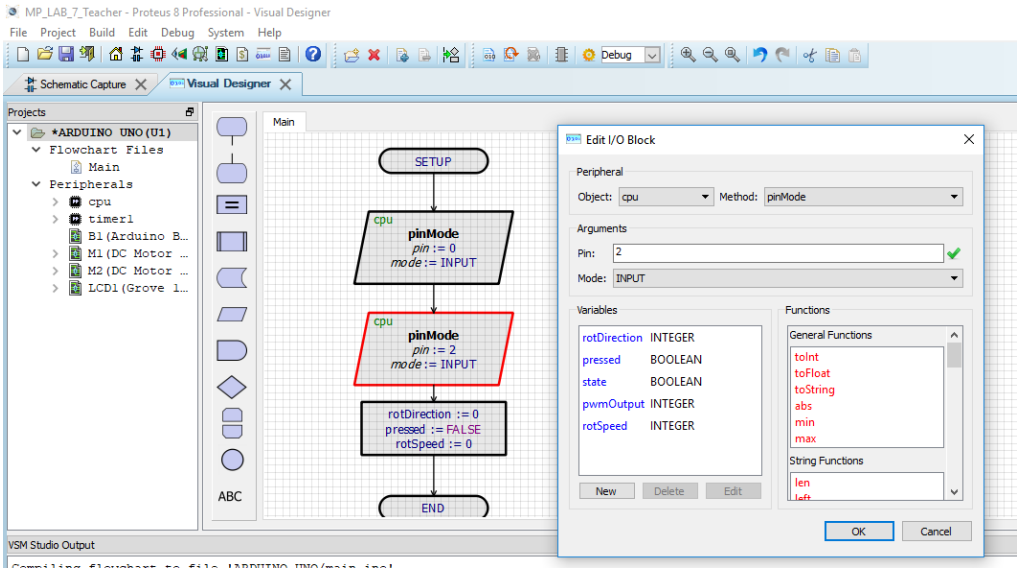
Flowchart for Proteus Visual Designer:



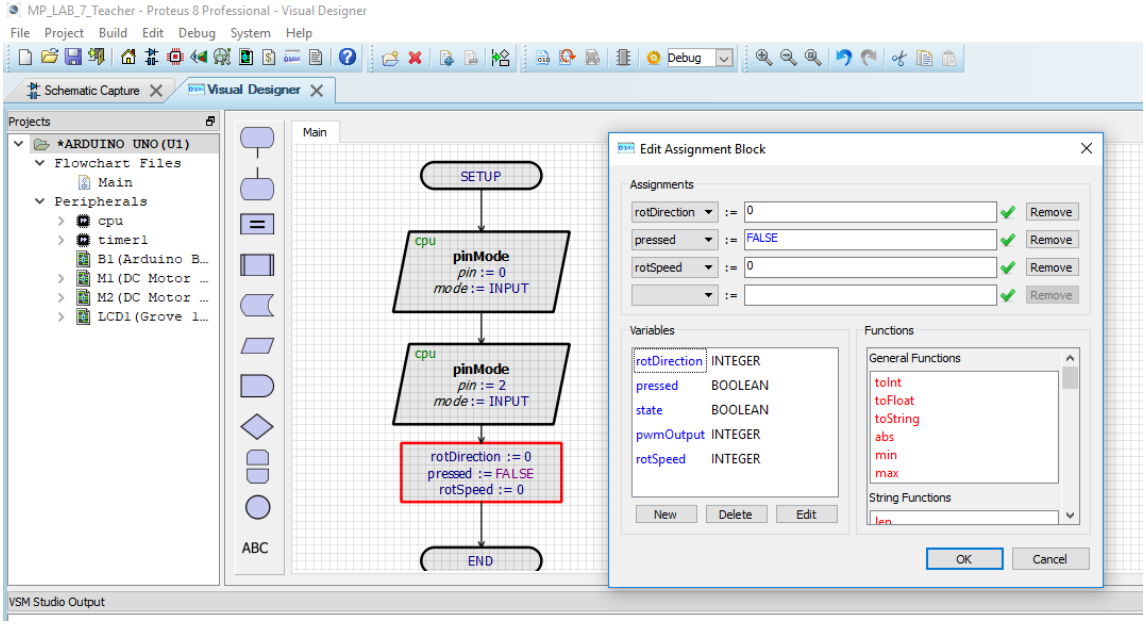
For creating your algorigthm, connect event block and name it as SETUP, to two I/O (peripheral) operation blocks and then with one assignment block.

Next double click on each block and edit the Blocks as below for specifying the inputs :

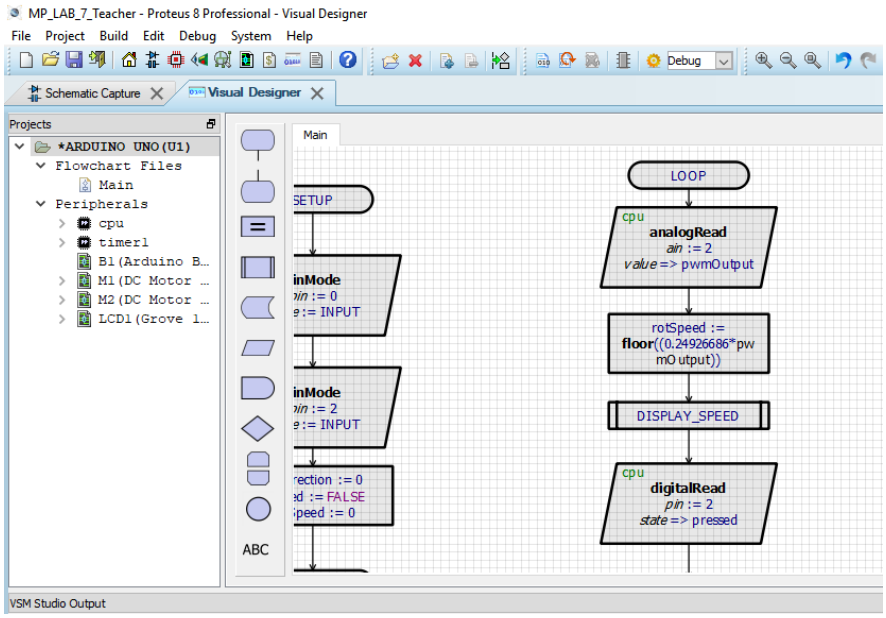




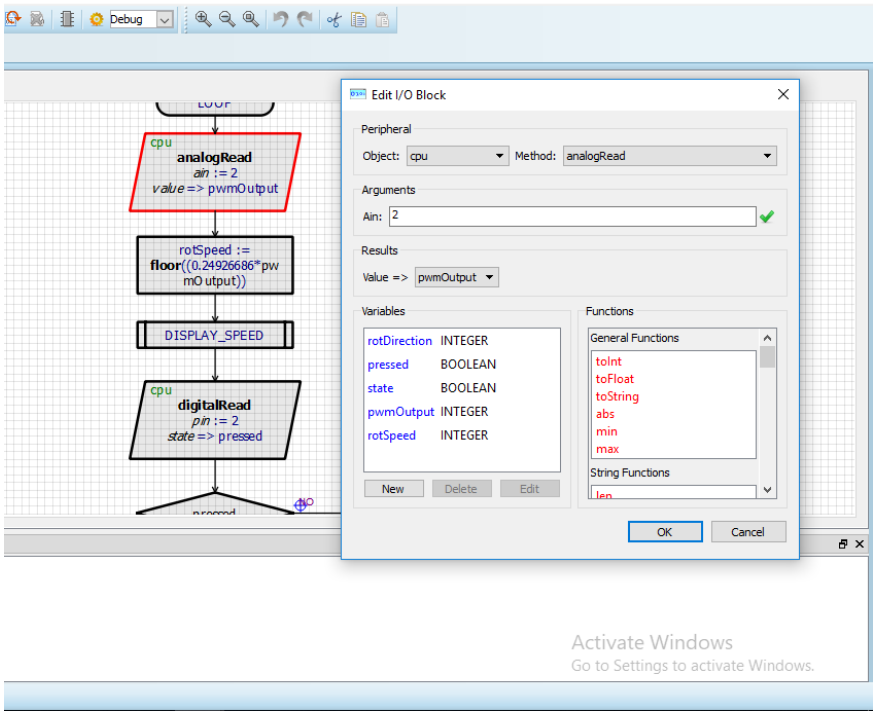
Set up the initial conditions like diection and rotational speed of motor as 0 and set the assignments for variables declaration too.



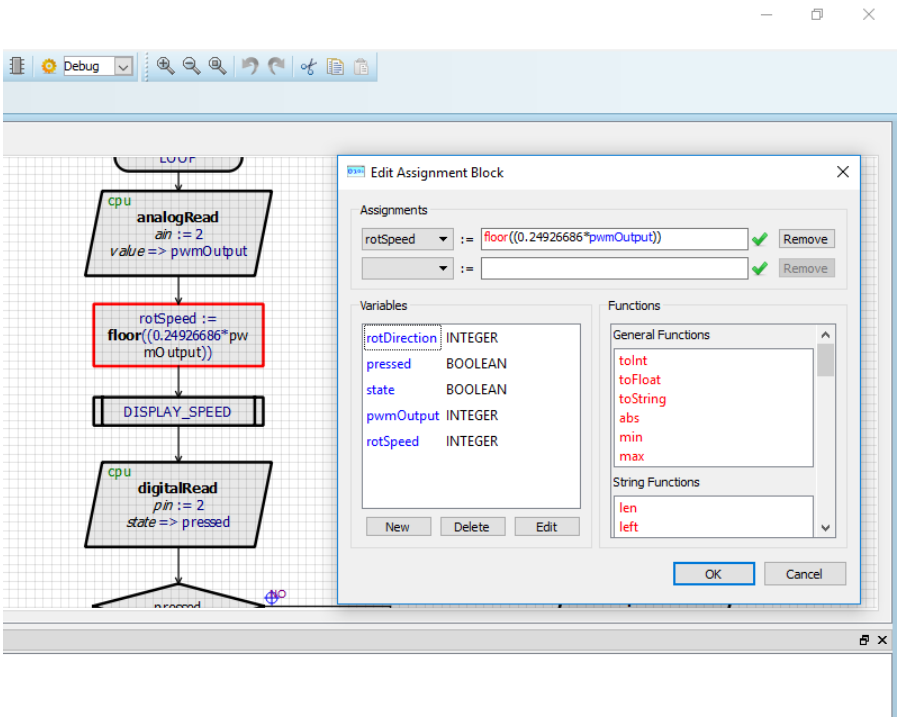
Next task is to create the loop function for setting up the speed of rotation of motor M1, then to display the speed through OLED display and also specify the value for POT-HG along with adding conditions of the button while being pressed or unpressed.



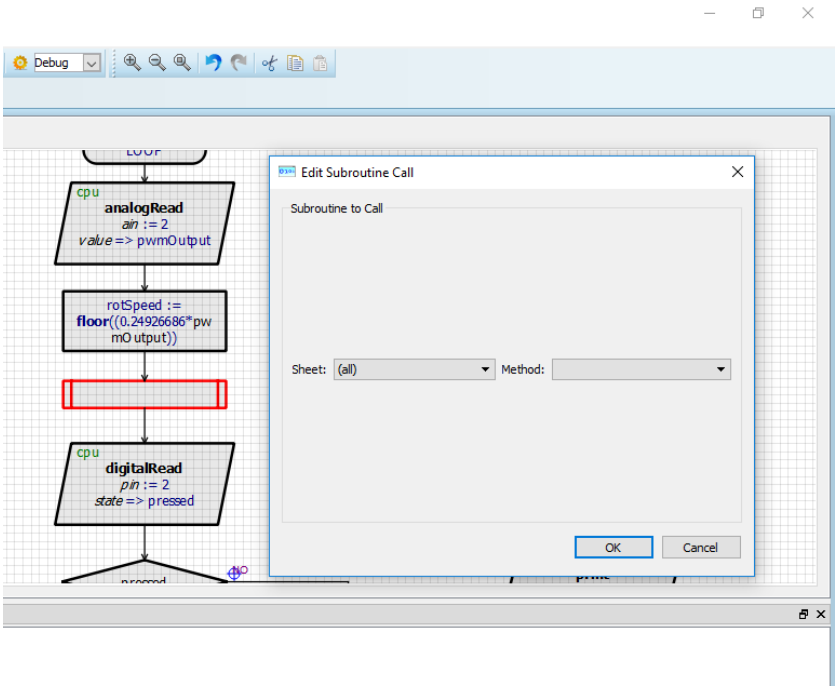
Edit the analogRead block as below as well for starting the loop process.



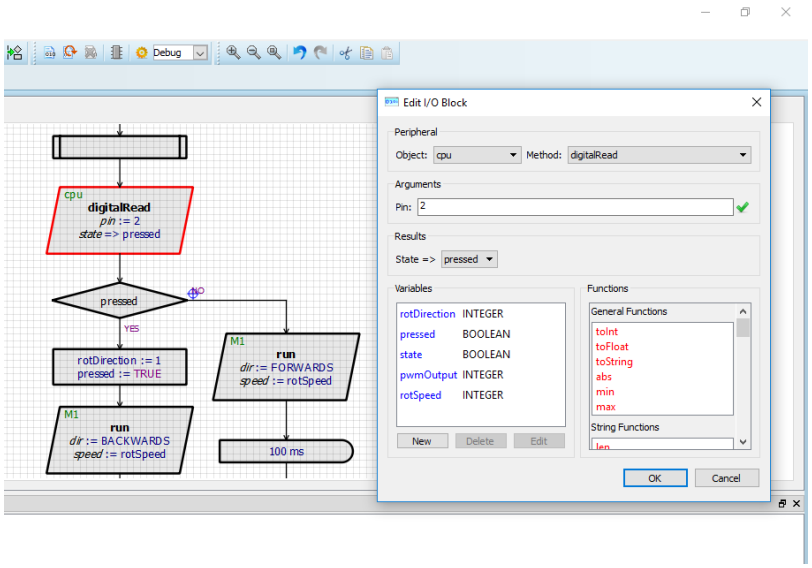
Drag one assignment block to write a formula,



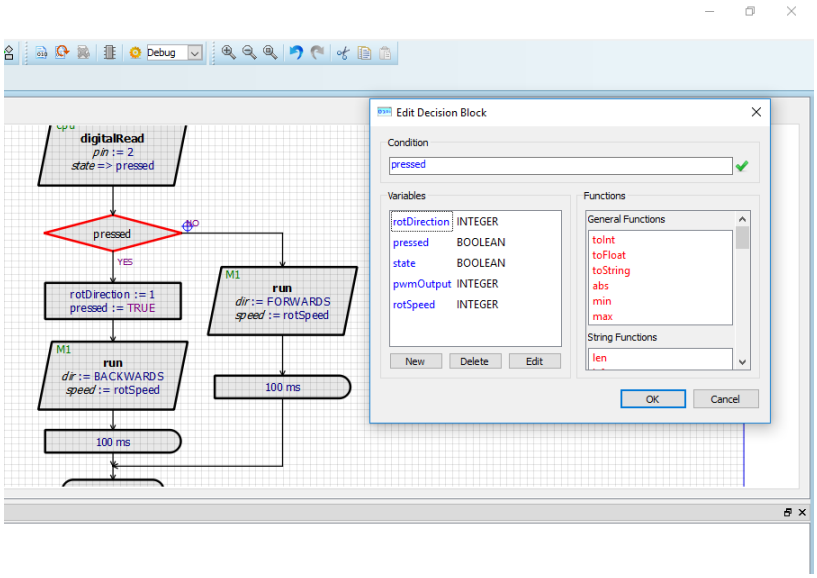
Add one subroutine call block for future work and edit the block as well.



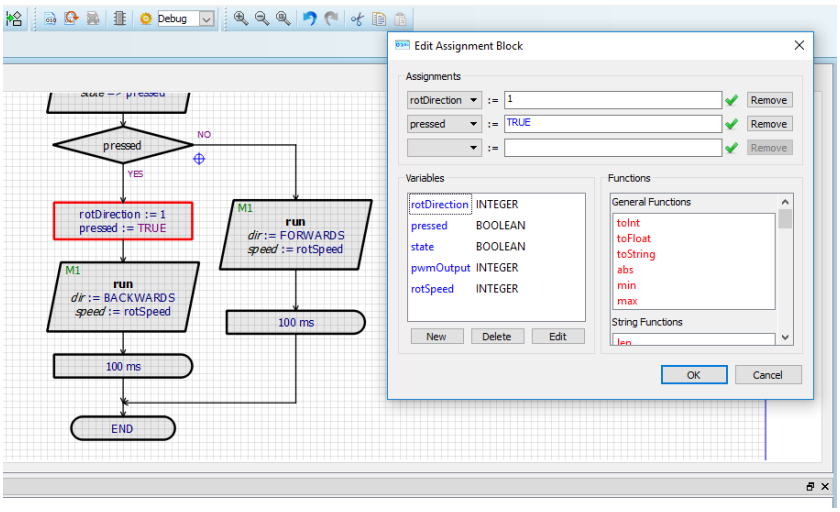
Now edit the digitalRead block as well considering pin 2 .



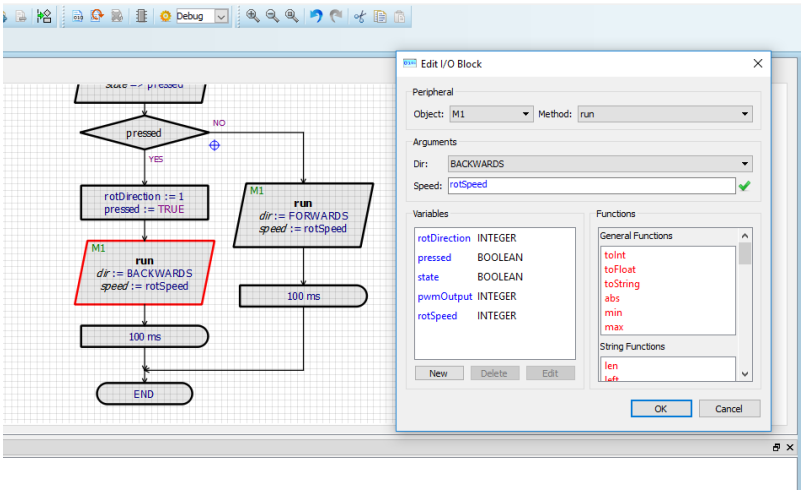
Add a decision block for the push button state to control the direction for running the motor M1



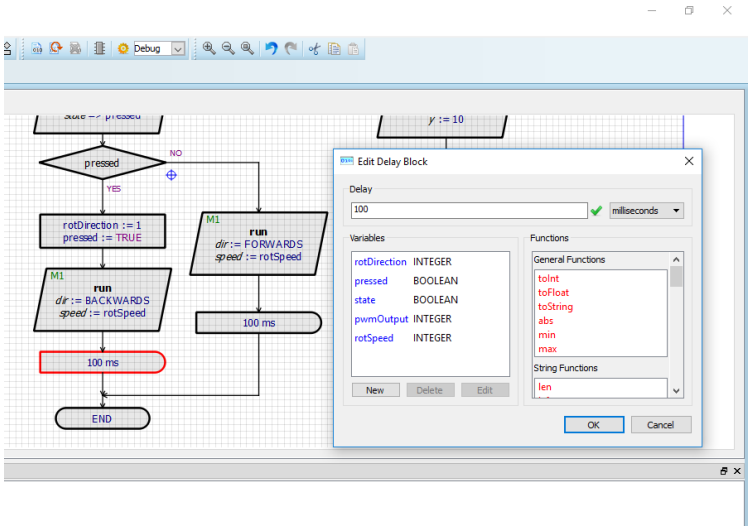
To activate a backwards direction of rotation , add one assignment block and edit as belows.



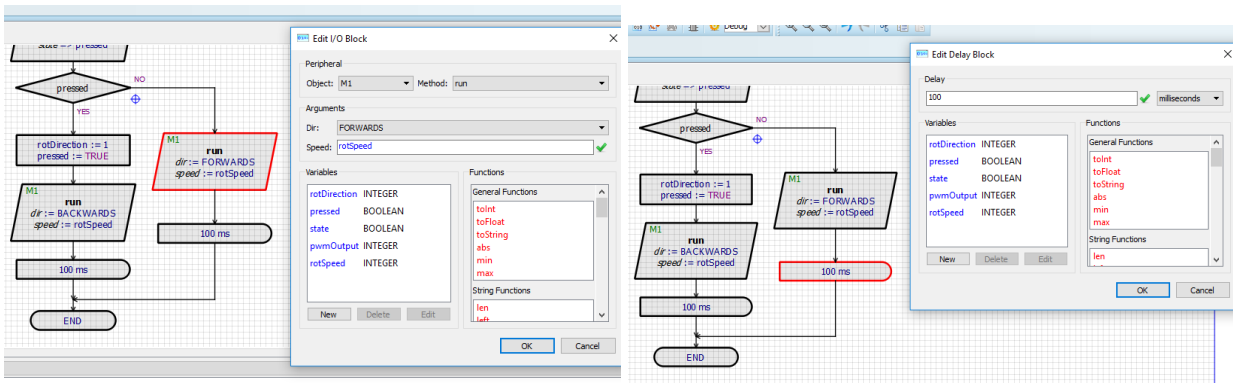
Select your object as M1 (motor), method as run, direction as Backwards and speed as rotSpeed as below.



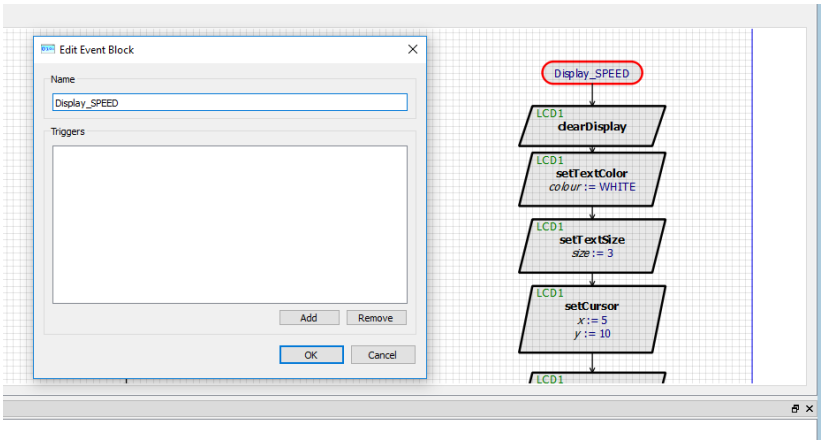
Include a delay of 100ms using time delay block as below.



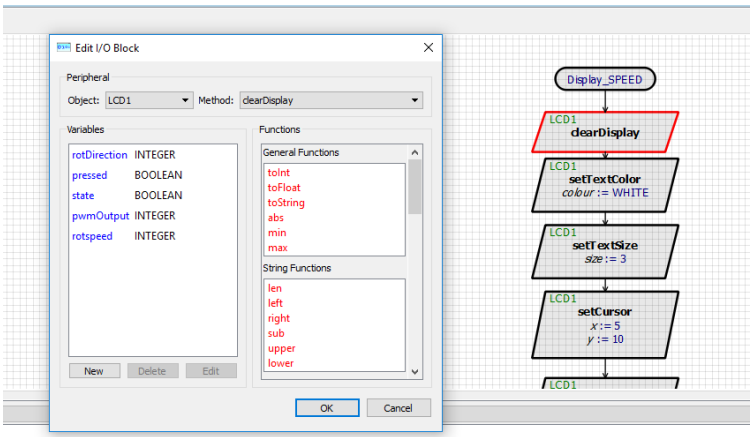
Edit the blocks in same manner for forward direction also along with setting the time delay and connect the dots with end process for loop.



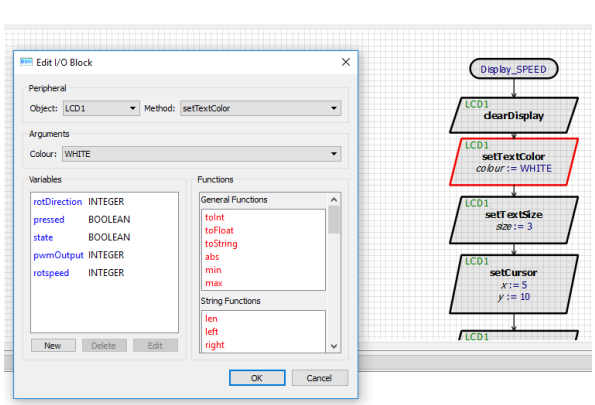
Now, the LCD display component should be activatedusing another Event block and naming it as Display_SPEED.

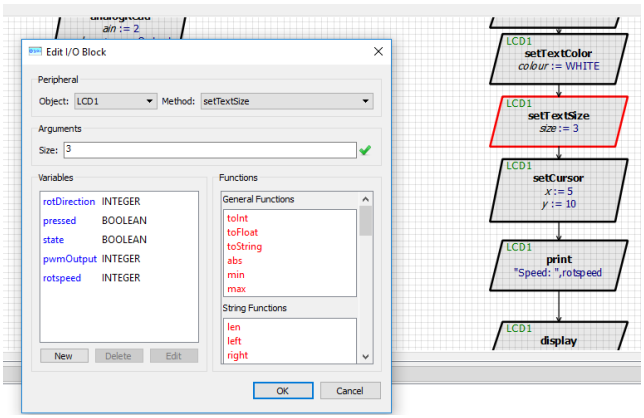


Add a block for adding LCD1 as an output component and select clearDisplay as method.

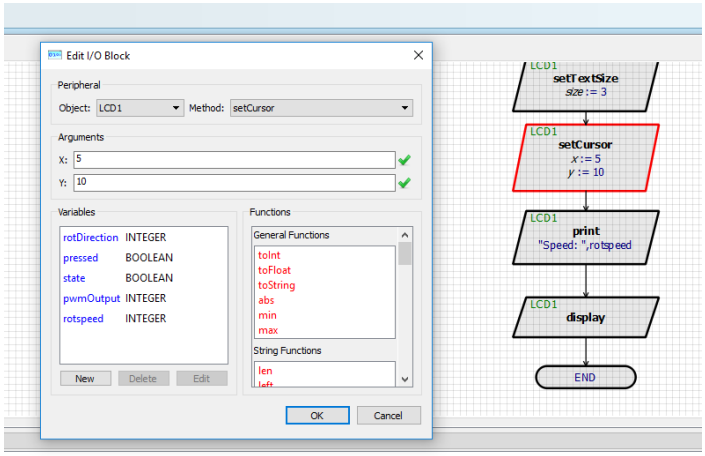


Select a color for text (here white is selected) and a size (here 3 is selected) for text as below.

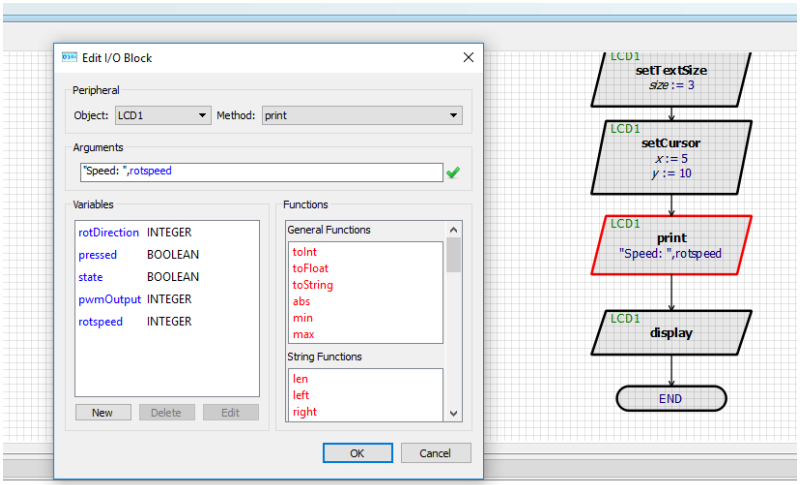




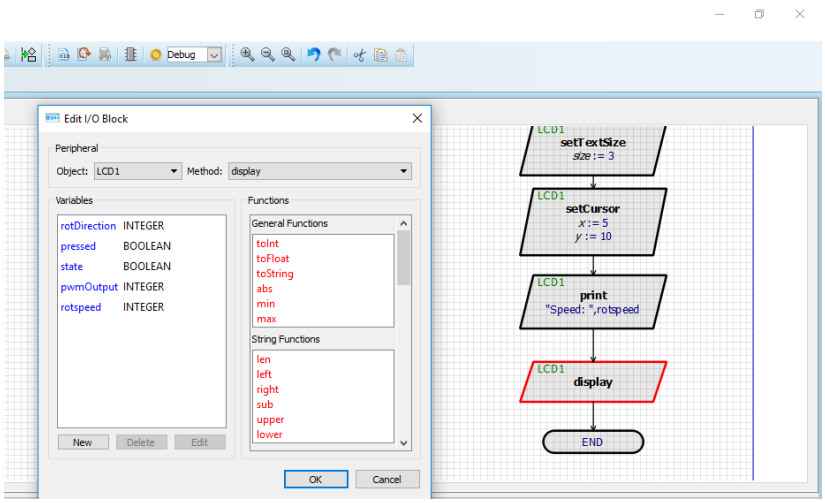
Set the cursor also by editing the block as below:



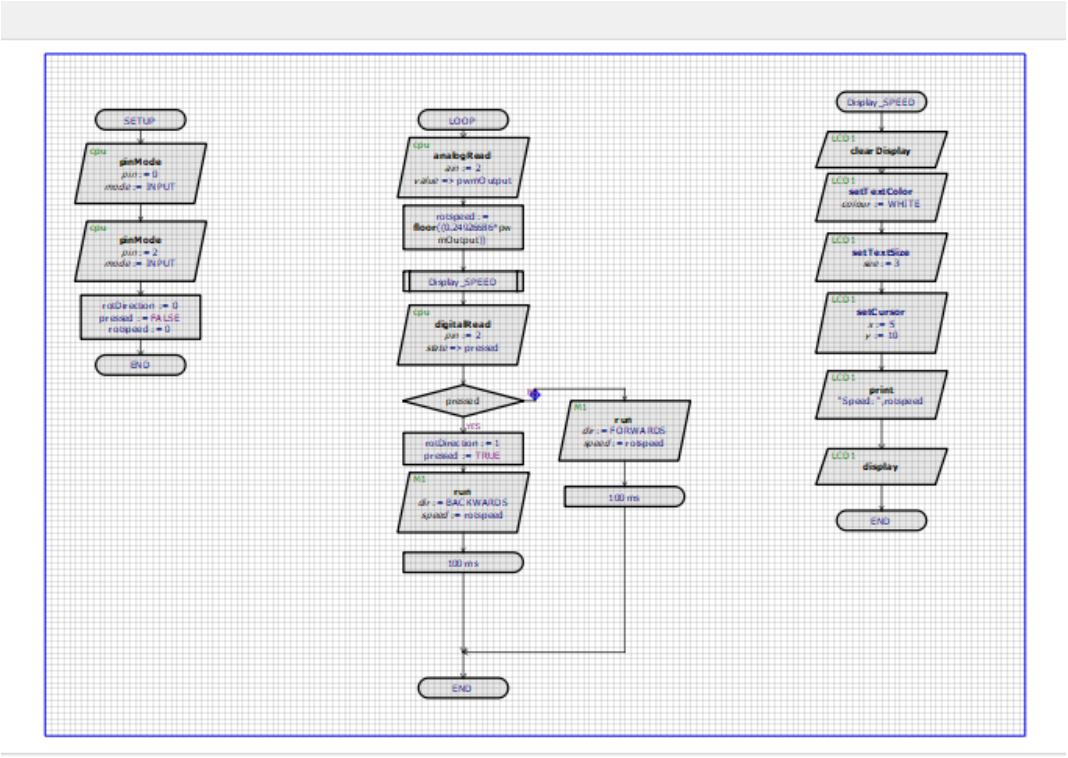
For printing the output edit the block as below and select “Speed” , rotorspeed as arguments.



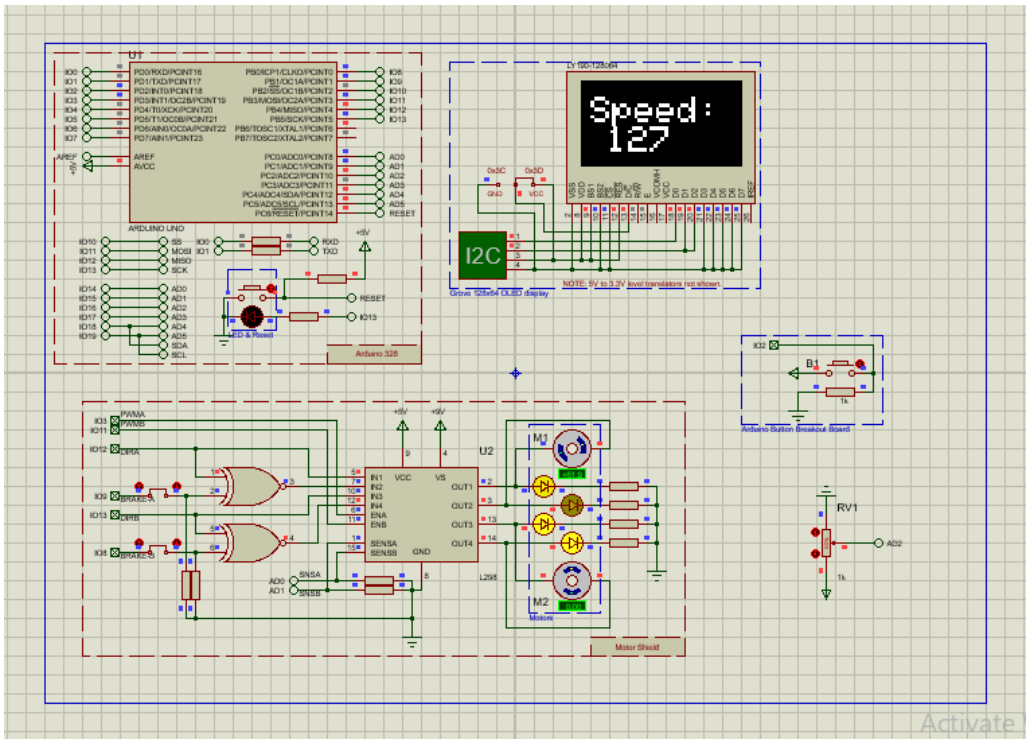
Activate the LCD display by editing as below and end your overall algorigthm process.



Thus the overall algorigthm would look like as below:



Next build the project and click the play button in schematic capture. For duty cycle of 50% , motor speed and LCD output is shown below:



Thus, speed in LCD display along with speed of M1 motor (sign is +) can be varied by increasing or decreasing the percentage of duty cycle using POT-HG. When push button is pressed, the direction of Motor M1 gets altered with same speed (sign is minus(-)) .

Note that under the Motor Control heading there are actually two shields. These are identical except that the first has the board configured and populated with two DC motors and the second is configured and populated with one stepper motor. In the real hardware of course you would need to configure and populate the shield manually according to which project you were programming the Arduino with.

You should notice that in the Project Tree you now have two motors with associated methods.

As always, you can program the physical board at any time via the upload button.

Questions for report writing:

1) Include all codes and scripts into lab report following the writing template mentioned in appendix A of Laboratory Sheet Experiment 1.

Reference(s):

1) https://www.arduino.cc/.

2) https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/