Lab 1 – Feasibility Model Phase 1

ECE 298 – S2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Members** | Team Member 1 Barry Cimring  Team Member 2 Anweshi Avadya | **Team Number** | 90 |

# Part 1 – Project Design Requirements

1. **Functional Requirements**
   1. The speed and direction of both wheel must be controlled by two (2) potentiometers acting as speed and steering inputs. The result of the potentiometer movement must correspond to max/min speeds and max/min turning directions.
   2. The motorized wheelchair must be able to rotate its wheel at a maximum speed of 1.67 m/s and minimum speed of -1.67 m/s when moving forward, which is approximately the speed of walking human adults.
   3. …
2. **Non-functional Requirements**
   1. The ramp-up of the motor controlling the speed of the wheelchair must be pleasing to the user of the wheelchair.
   2. The bulkiness of the hardware components of the project must be small enough such that the user will not think it stands out on the wheelchair.
   3. When purchased, the ease of setup must be such that the user is required to perform as minimal operations as possible (i.e., it is fool proof).
3. **Constraint Requirements**
   1. The total cost of the wheelchair control system must not add more than $50 to the total cost of the wheelchair.
   2. The wheelchair control system must not be more than 0.5 kg to keep the weight of the wheelchair mostly the same.
   3. The wheelchair control system must be kept as simple as possible without any complexities that could hinder user experiences.

# Part 2 – Project Considerations for I/O

## **Project Sensors and User Inputs**

**Potentiometer**

The potentiometer will act as a user input to the control system that determines the speed and direction that the wheelchair must go. The range of the potentiometer should be from 10 k to 100 k which will be connected to an analog input pin of the MCU in a voltage divider circuit (more described below in ‘Potentiometer Connection’).

**Potentiometer Connection**

Both potentiometers will be connected as the first resistor in a voltage divider circuit powered by the battery. At the node between the potentiometer and the second resistor, one of the ADC IN pins will be connected. This second resistor will be on the order of 100 k to ensure that the minimum sensed voltage of the ADC is 3.3/2 V = 1.65 V and the maximum voltage is 3.3 V \* 3 V. This will correspond respectfully to maximum/minimum speed and maximum left/maximum right turning for speed and steering. To identify whether or not a change in the potentiometer has been made, the potentiometer’s change in input will trigger an interrupt that the MCU will deal with my changing the speed and steering direction of the wheelchair.

**Battery-Level Sensor**

A circuit will be connected to the external battery to measure its voltage level over time. It is assumed that the maximum possible battery that this project requires is 20 V. The battery-level sensor range must be able to transform a 20 V input voltage into a voltage on the range of 0-3.3V as input to the MCU’s ADC.

**Batter-Level Sensor Connection**

The circuit that will sense the battery level, whose maximum possible value is assumed to be 20 V, will be connected to the ADC peripheral device of the MCU. The voltage of the battery will be sent through an op-amp buffer to isolate it from the ADC. The output of the voltage buffer will then be connected to a voltage divider whose output range (for an input range of 0-20 V) is 0-3.3 V, the maximum voltage of the ADC. This can be achieved with a R! and R@ ohm resistor.

**DC Motor Encoder**

The motor encoder that will be used in this project is also part of the motor. The motor encoder outputs 3 square waves whose frequency is dependant on the speed of rotation. The output labelled ‘IDX’ send a pulse whenever the zero value of the wheel is rotated around (i.e. a revolution has occurred). The two outputs labelled ‘Q1’ and ‘Q2’ will determine the angular speed (which can be forwards or backwards) and the absolute angular position.

**DC Motor Encoder Connection**

The outputs of the motor encoder (IDX, Q1, Q2) will be sent as inputs to three of the GPIO pins of the MCU. A counter will be used to time the rate at which IDX pin transmits a pulse (to determine rotational speed) and a SOMETHING IDK YET will be used to determine which of Q1 or Q2 is high while the other is low to determine the direction of rotation.

**Switch**

The switch used in this project will be required as a user input for a couple of different requirements. They will be used as an On/Off button and they will also be used to switch the mode of operation between Locked or Run Mode.

**Switch Connection**

## Project Actuators and User Outputs

**Motor + Motor Encoder**

An actuator required for the project is a DC motor that will control the speed of rotation of the wheelchair’s wheels. It is assumed that the number of revolutions of the motor to the number of revolutions of the wheelchair’s wheels is 6:1. Thus, as per the functional requirement, the maximum/minimum rotation speed of the motor must be (if we assume that the wheel is of 0.25 m radius): . Therefore, the max/min rpm of the wheel is .

**Motor + Motor Encoder Connection**

Because the DC motor requires a negative voltage to spin in reverse, the motor will be connected to an TIMx pin of MCU via a series of op-amps. The TIMx pins are channels that are able to output a PWM signal, the duty cycle of which will control to speed of rotation of the motor. The first op-amp will reverse the polarity and reduce the value by a factor of two of the 3.3 V signal from the MCU. The second op-amp will add the voltage from the previous op-amp to the PWM signal from the MCU from 0 to 3.3 V. This puts the voltage range of the input to the motor at -1.65 – 1.65 V. In Part 3, when testing the motor we see that this is enough range to achieve the maximum/minimum speeds desired.

**Multicoloured LEDs**

The LEDs are used as user outputs to convey import information. The convention the LEDs follow are that when the wheelchair is in the ‘run’ mode, a green LED is turned on. Another set of LEDs are present to convey the battery level of the controller. If the battery is greater than 90% another green LED is switched on. If the battery charge is anywhere between 80%-90% then a yellow LED is turned on. If the Battery level is anywhere between 60%-80% then an orange LED is turned on and if the battery level is below 60% then a red LED is set to be flashing. Each setting of the LEDs is created to convey a specific message to the user.

**Multicoloured LED Connection**

**LCD – Liquid Crystal Display (LM016L)**

The liquid crystal display (LCD) will be used to display the that the wheelchair is in and the RPM of both wheels on the wheelchair. The display will show 16x2 characters. The top row will display the mode that the wheelchair will be in (Locked or Run mode), and the bottom row will display the RPM of the left and right wheels.

**LCD – Liquid Crystal Display Connection (LM016L)**

The LCD has 11 digital inputs, power (VDD) and ground (VSS). Only 8 of the digital inputs will used as the LCD will be used in 4-bit mode to minimize wiring connections. The digital inputs on the LCD are the RS, RW (read/write), E (enable) and D[4..7] pins. The digital output pin will only be to D0 pin that indicates if the LCD is busy with BF (busy flag).

The LCD will be initialized and controlled via a commands as described in the LM106L datasheet (examples provided in section 3). To achieve this, the pin of the LCD’s I/O pins will be connected to 8 of the 16 GPIO pins on the STMicro MCU. The STMicro will send initialization commands and read/write commands in 4-bit format.

## Project MCU Internal Resources

**ADC – Analog to Digital Converter**

The STM32F401RE has a built-in ADC peripheral. The ADC will be used to measure the voltage level of the battery as well as the voltage level from the variable resistor voltage divider that indicates the speed and direction that the user desires.

The ADC, which can have 16 inputs, can be read via channels that the user can set in code. The ADC can be triggered in code, sending its value to a memory address, then read from at 12-bit accuracy on a 0-3.3 V scale (0xFFF indicates 3.3 V, 0x000 indicates 0 V).

The time it takes for the ADC to read from an MCU channel and send to memory is approximately 9 us. Given this application’s requirements of reading from a human-input and battery voltage level, this will be fast enough not to require the direct memory transfer that the DMA can facilitate.

**Internal MCU timer**

**GPIO**

**Interrupts**

External interrupts will be necessary to keep the MCU as free as possible to complete other tasks. The external interrupts will be used to indicate whether the voltage level of the battery has dipped below a certain threshold or the variable resistor has been changed by the user of the wheelchair. This prevents the MCU from polling each of the two input signals periodically to receive input.

Additionally, interrupts will monitor the input pulses from the motor encoder and measure how many pulses the MCU receives per 0.5 seconds, as well as which direction the wheels are rotating via a comparator circuit. An internal timer interrupt will then compute the instantaneous RPM (averaged over 0.5 s) and output the value and direction of rotation onto the LCD.

**Software Parameters**

The software parameters that must be kept track of are:

1. The voltage level of the battery.
2. The angular speed of the two motors attached to the wheels.
3. The angular direction of the two motors attached to the wheels.
4. The desired user input (after processed from the potentiometer input) of the speed of the wheelchair.
5. ANYTHING ELSE

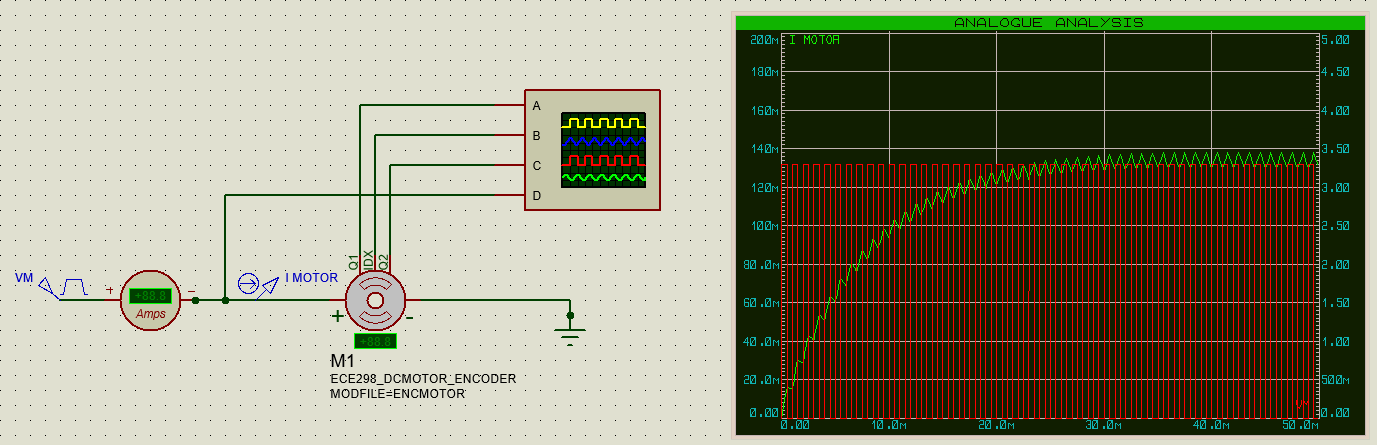
# Part 3 – Device Testing Methodology

## Device 1 – ECE298\_DCMOTOR\_ENCODER

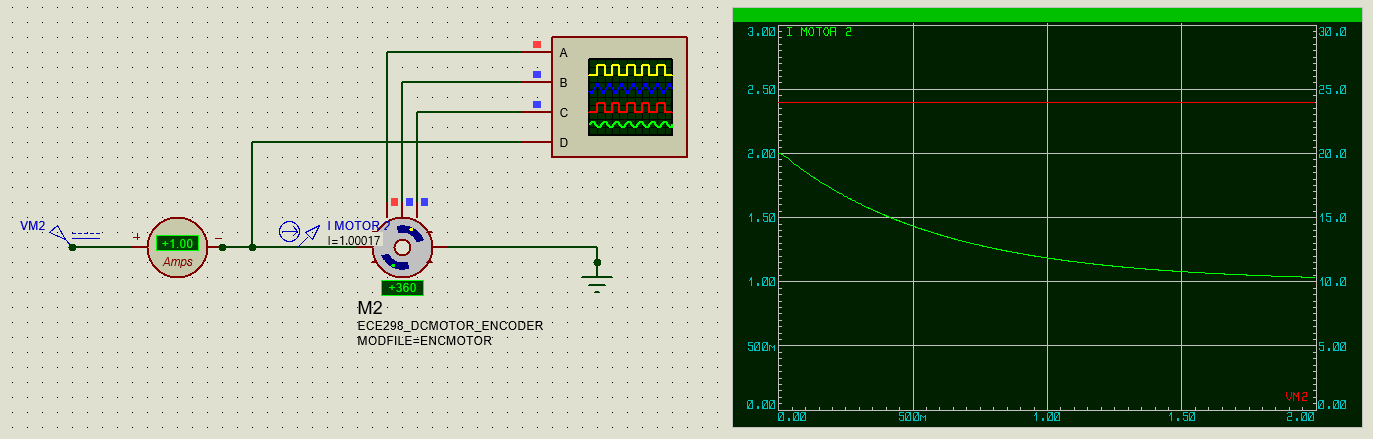
The ECE298\_DCMOTOR\_ENCODER is the DC motor that will be used to control the speed of revolution of the wheels on the wheelchair. This motor’s rotation speed is controlled by a DC current, and the current that the motor draws from a power source is proportional to the rotation speed of the motor.

To control the device, a voltage source will be connected to the + pin and the – pin of the motor will be connected to ground. For testing this is shown as a voltage source but in reality, this will be a pin that will output a PWM signal from the MCU timer.

The schematics used to test the DC motor is below:

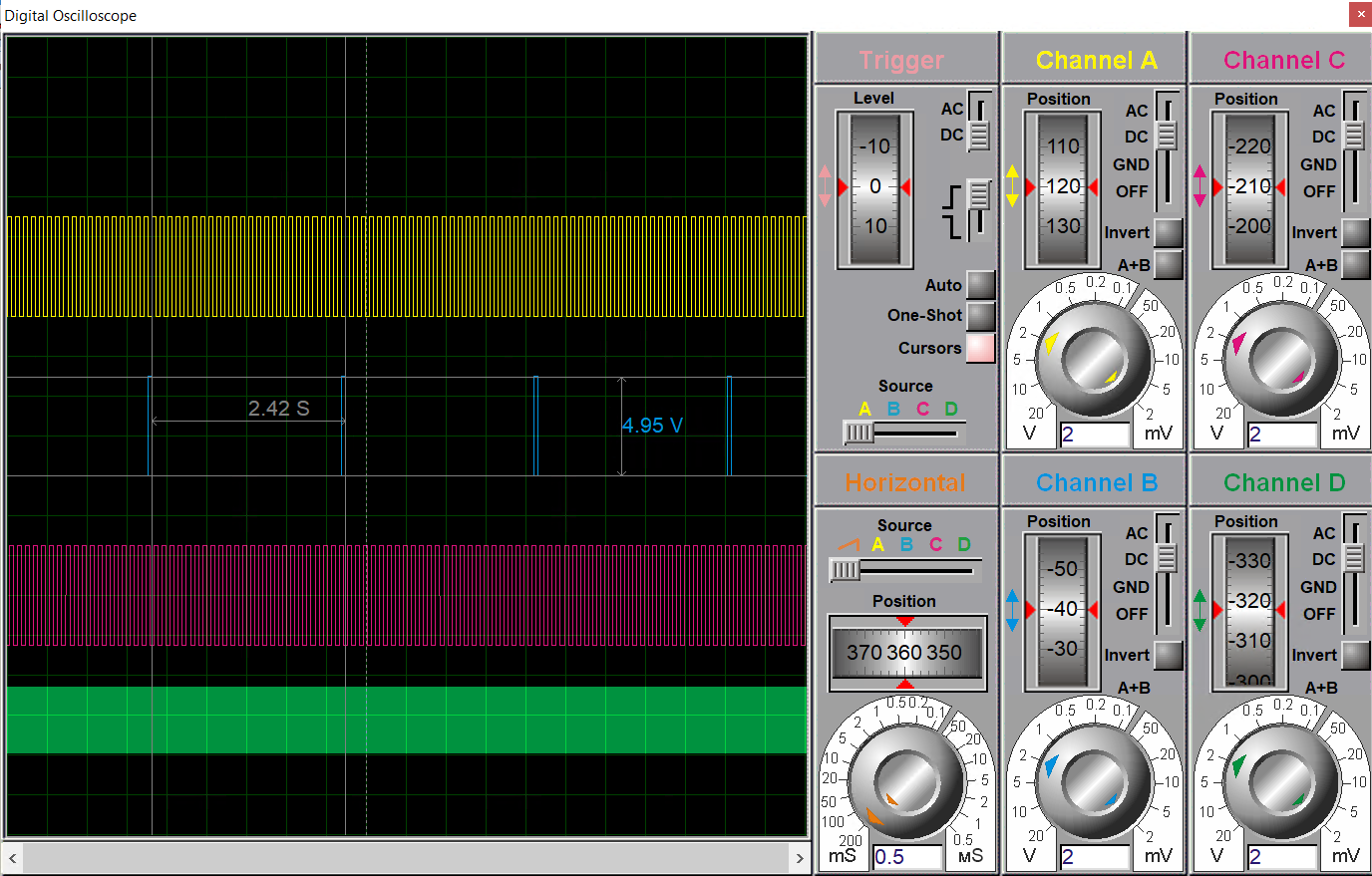
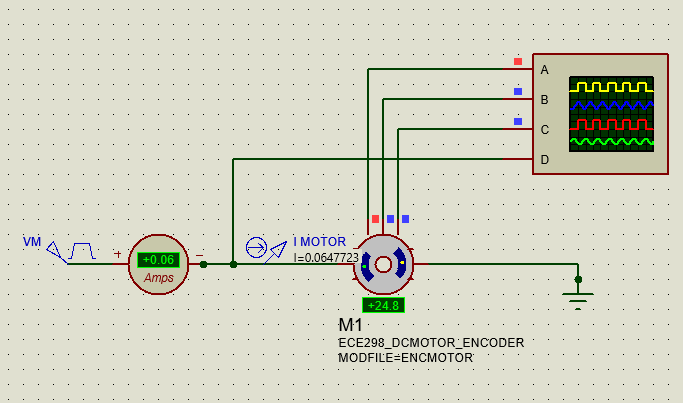


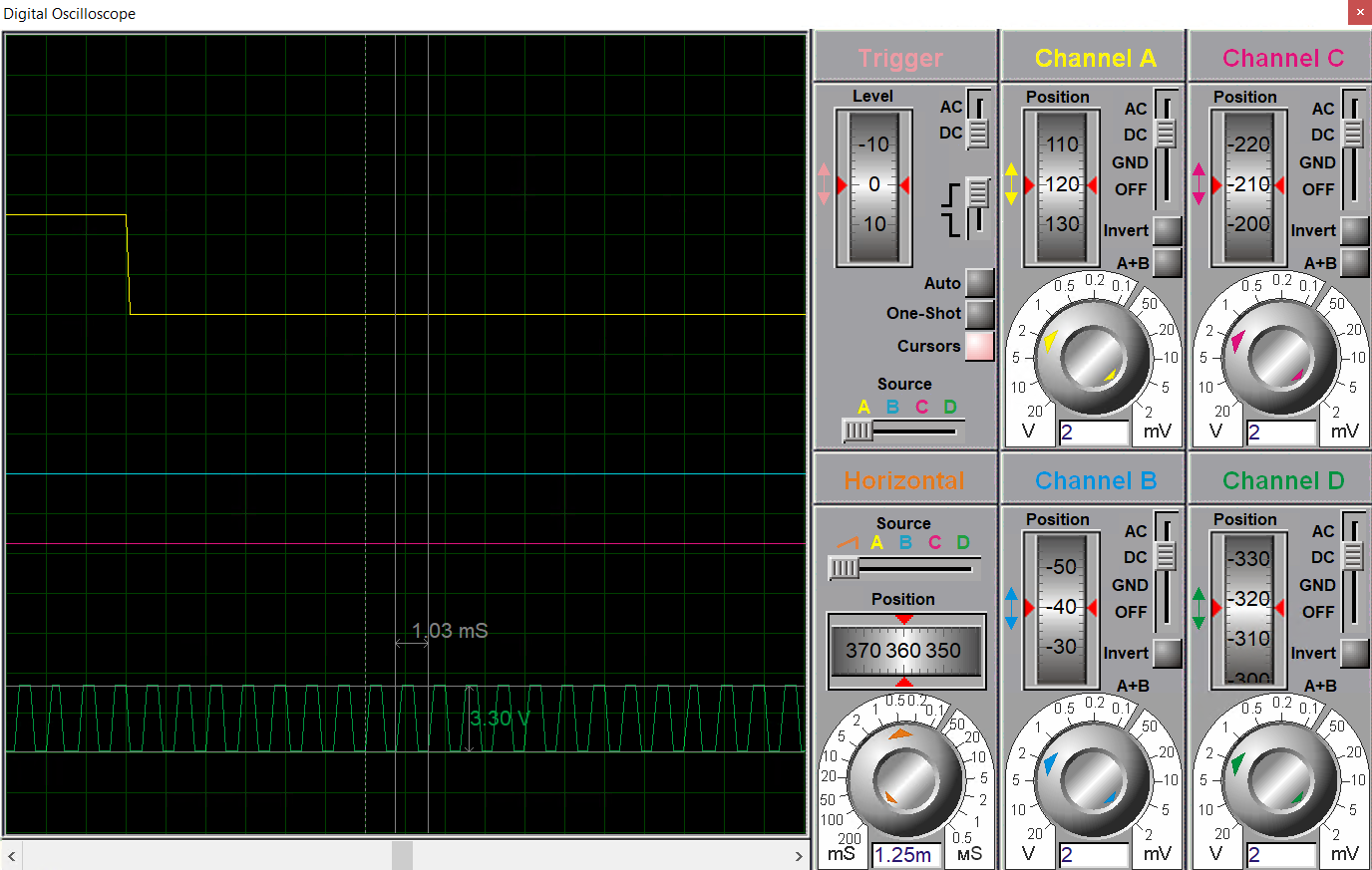
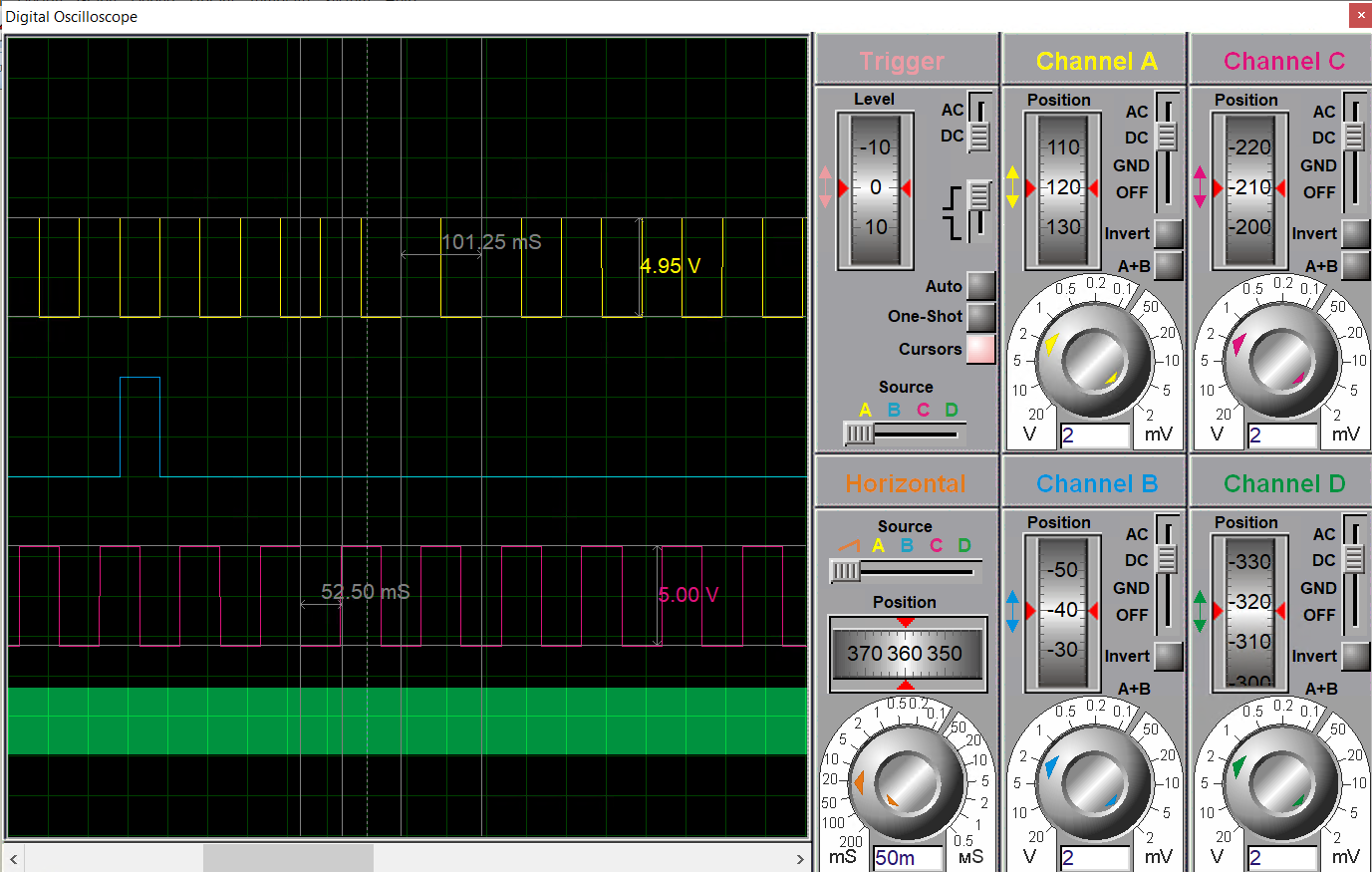
The DC motor’s positive pin is connected to a PWM signal which is at 1 kHz, V\_high = 3.3 V and V\_low = 0 V, which draws a current. In the figure above it is shown that when sent a PWM signal there is a rise-time of the motor’s current, and therefore rotation speed. This is estimated to be 30 ms at the PWM sent. When sent a DC signal of 24 V is applied to the motor:



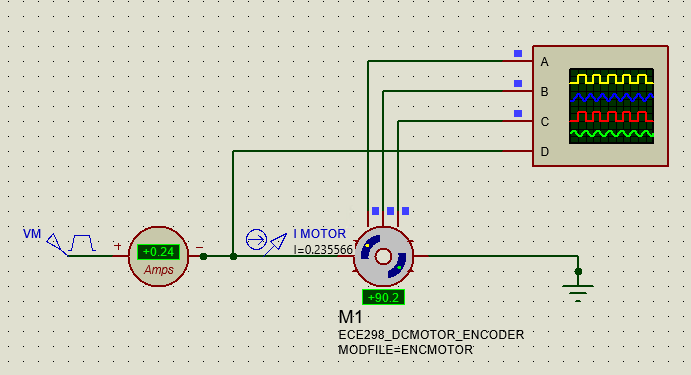
The current initially starts at almost 2 A and decays to its average value of 1 A. This is quite a lot of power. The decay is expected behaviour as the motor can be modelled as an inductor in series with a resistor.

The output Q1, Q2, IDX pins, as well as the motor voltage are connected to an oscilloscope to monitor the output of the motor. It is seen that the rotation speed (in RPM) of the motor is directly proportional to the voltage *and* the duty cycle of the PWM signal supplied, and hence the current. At an average current of 0.10 A, the motor’s rotation speed is 37 RPM, and at 0.24 A the motor rotates at 90.2 RPM. Below is an oscilloscope capture of the output pins for the same schematic:

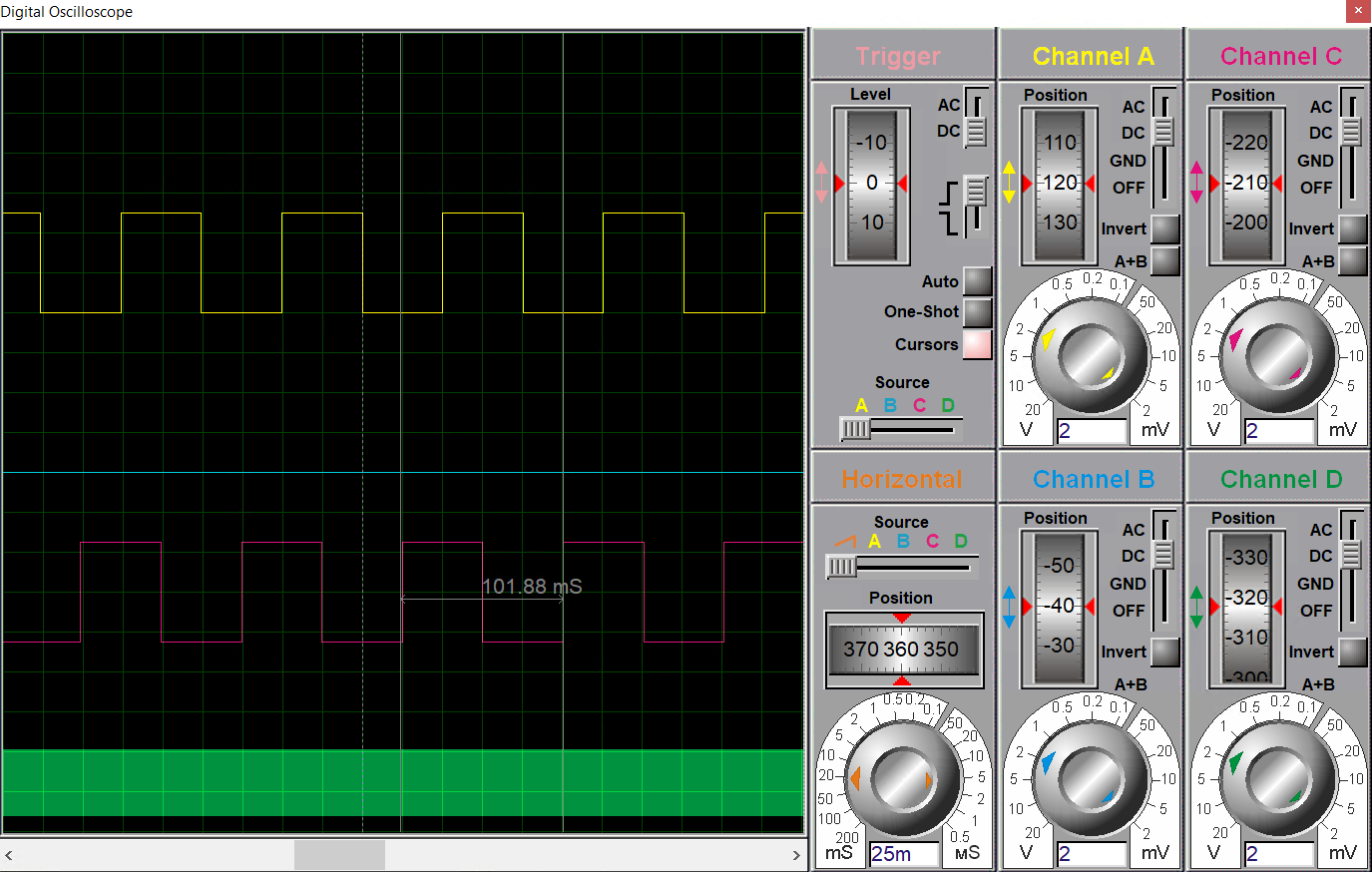
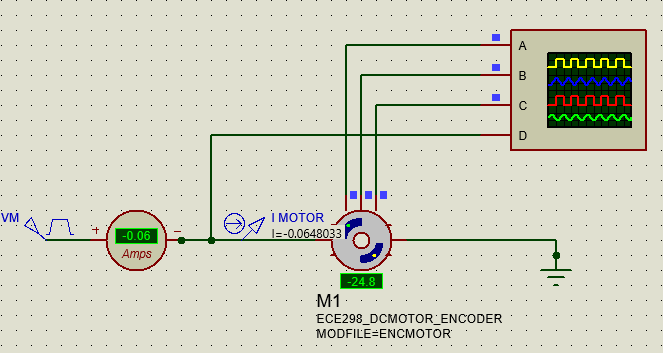




As shown, for 0.06 A input (on average), where the PWM signal is a 50% duty cycle wave with 1 us rise and fall time, the output of Q1, Q2 is a square wave from 0-5 V with a period of 101.25 ms. This period is inversely proportional to the rotation speed. The duty cycle of the square wave is 50%. When the motor is rotating in the forward direction Q1 turns from low to high when Q2 is low. The period of the pulse given from IDX indicating that a revolution has occurred is 2.42 s. The square PWM signal is also shown, with its expected value of 3.30 V maximum and 1 ms period. Below shows the state of the motor when a 12 V PWM signal at 50% duty cycle is given to it as input. It is seen that the voltage increases by 12/3.3 = 3.64, and so does the RPM of the motor 90.2/24.8 = 3.64. The periods of outputs IDX, Q1, Q2 are also reduced by the same factor.



We see that, in the same way, the motor will rotate in the opposite direction with the same magnitude of RPM if a negative voltage is applied: (a PWM signal of V\_high = 0V, V\_low = -3.3 V and 50% duty cycle is applied)



The same angular frequency occurs in the negative direction when the negative current is applied, and the rising edge of Q1 occurs when Q2 is high.

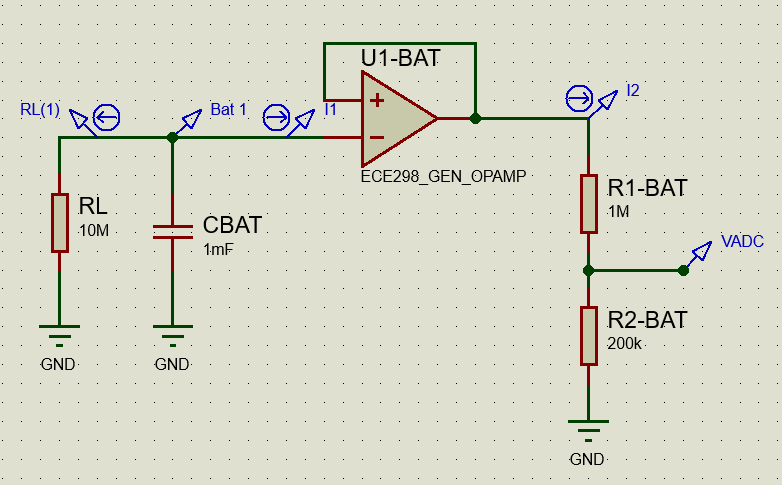
This motor has the output built-in to it so the MCU can receive input directly from the same component. Both the motor (actuator) and the motor encoder (sensor) has been tested in this section.

## Device 2 – Battery-Level Sensor

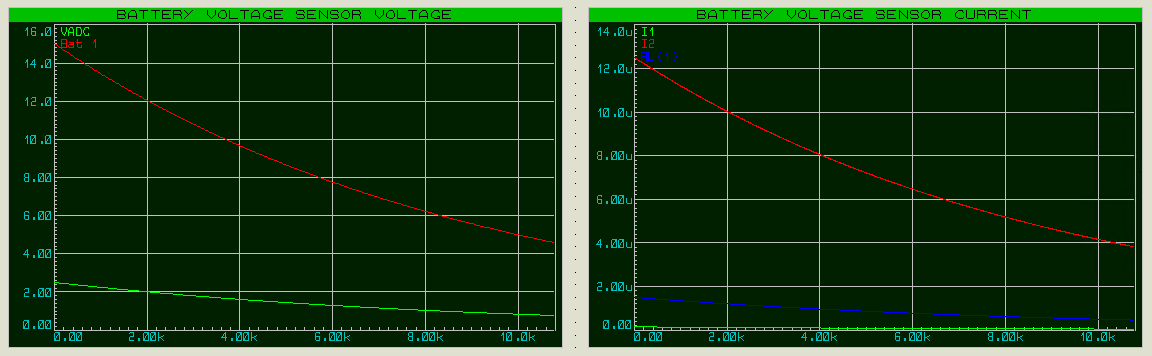
The battery-level sensor circuit is designed to output a voltage between 0-3.3 V given an input range of 0-20 V, which is the assumed maximum voltage of an external battery for this application.

The output of this circuit will feed into the ADC peripheral of the STM32F401RE which will be converted to a digital signal that the firmware of the MCU can read. Depending on the battery level, the MCU will determine which LEDs (indicating battery percentage) will be turned on.

A schematic for the battery-level sensor is shown below:



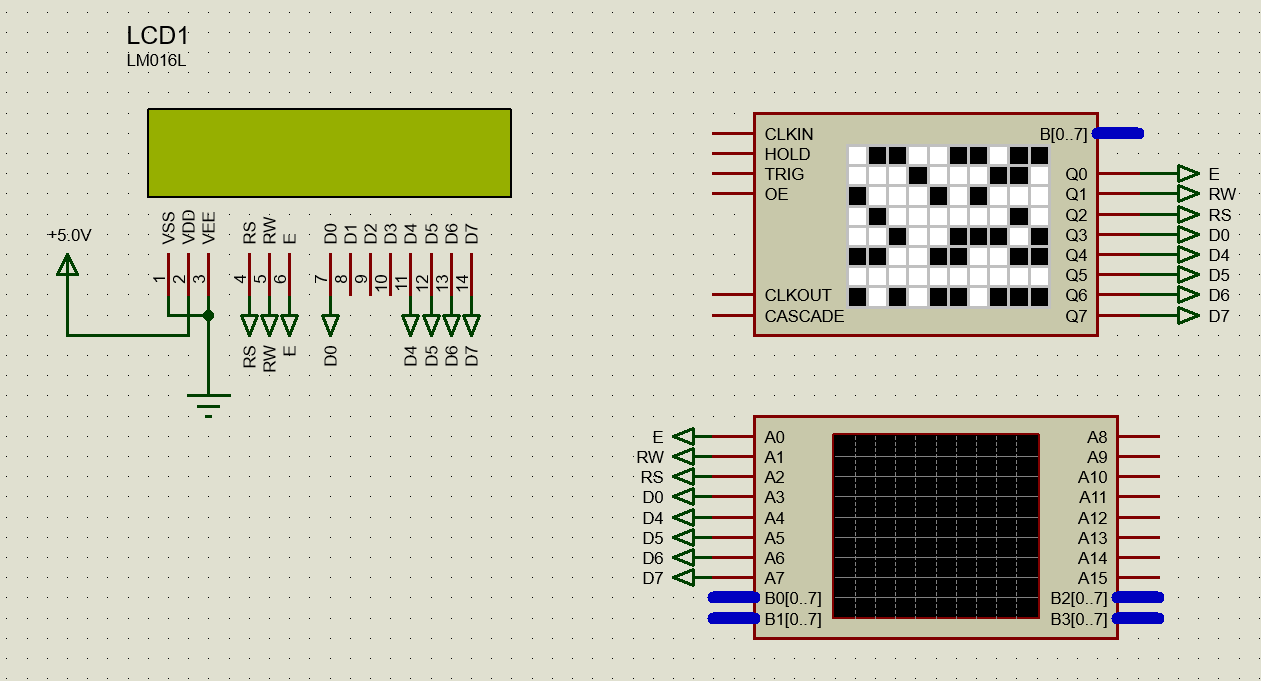
The battery is modelled by a large capacitor – CBAT – whose initial voltage is set to 15 V. The battery voltage is sent through a buffer to isolate the signal from the rest of the circuit (RL, representing the circuit load), which is the input of a voltage divider providing the correct maximum output rated for the MCU’s ADC. VADC will feed into the ADC of the MCU in the final implementation.

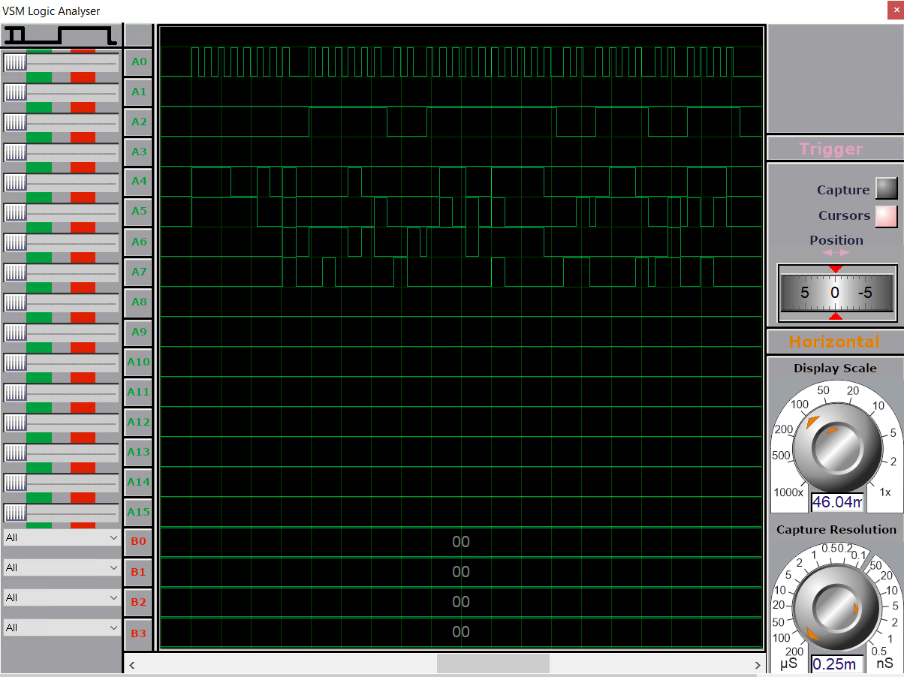
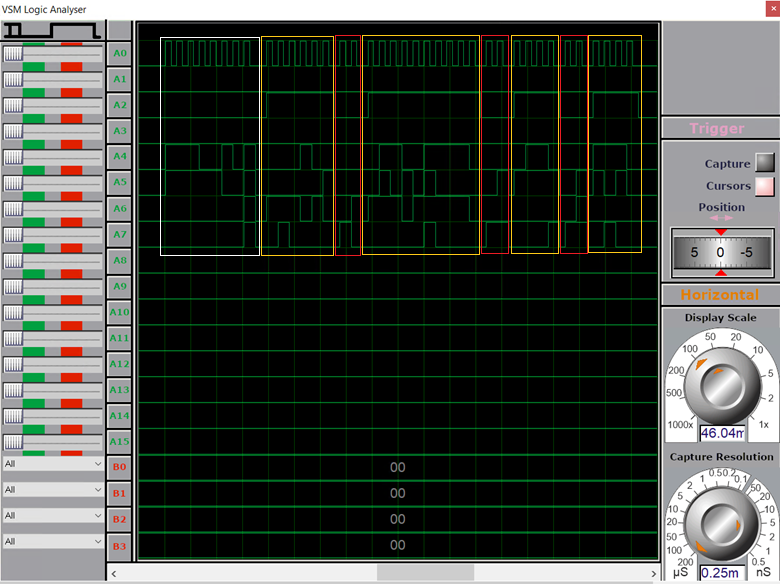
Due to the modelling of the external circuit as a large circuit, the battery’s voltage will drop exponentially over time (since the current drawn by the op-amp buffer is negligible). The lifetime of this battery is (rounded down). This is shown below in the time-domain voltage and current of VADC, Bat 1, I1, I2:

As shown the battery, in this configuration, retains at least 50% of its charge for 5.6k s = 1.5 h, which is exponentially decrease with the lifetime as stated above. The current drawn from the battery is negligible, and the current drawn in the voltage divider circuit is of the order of uA, which is also satisfactory. In this example the initial voltage read by the ADC would be 2.5 V – indicating to the MCU the battery is at 15 V charge.

## Device 3– LM016L

The LM016L is the LCD that will be used to display the RPM and mode that the wheelchair controller is in. This will be achieved by 8 GPIO connections to the MCU that will send data and commands to the LCD. Below is the testing circuit designed to ensure functionality of the LM016L:



The VSS and VEE pins of the LCD are tied to ground, while the VDD pin is tied to +5.0 V. The digital input pins RS, RW, E, D0, D[4..7] are connected to a digital pattern generator to simulate the GPIO pins of the MCU. Below is a screenshot of the initialization process sent from the digital pattern generator as measured by the digital logic analyzer:

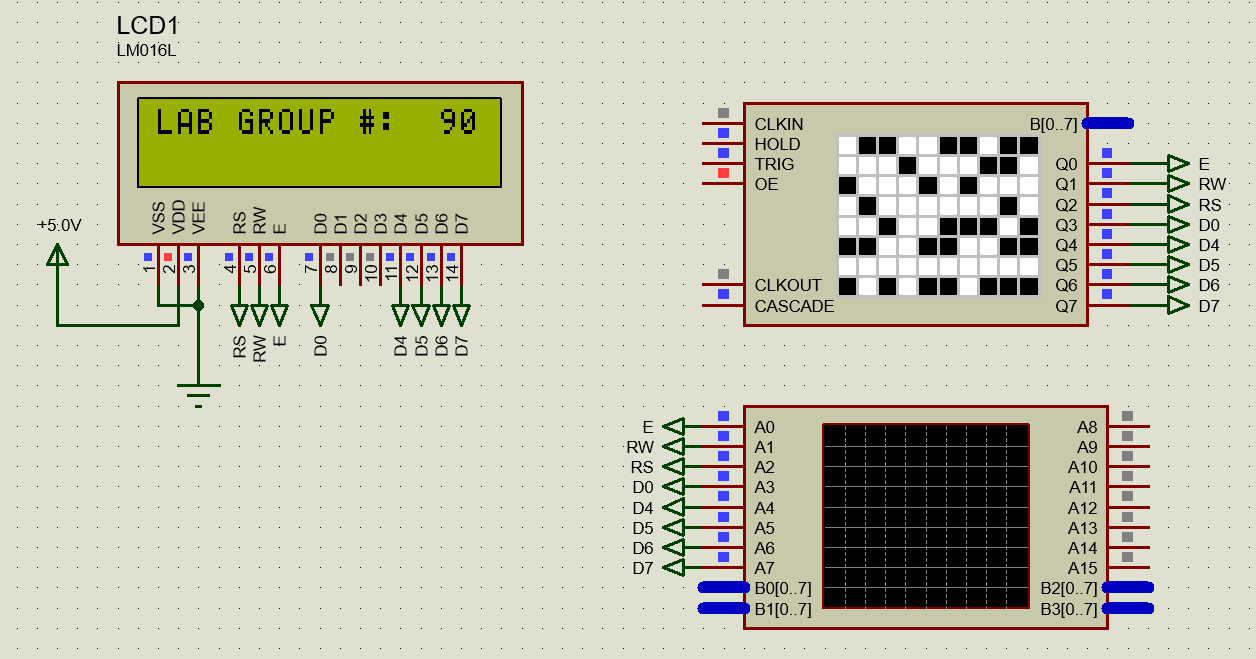
The pattern boxed in white represents the initialization procedure, the pattern in red represents changing the memory address of the LCD’s DDRAM where character data is written to, and the orange boxes show character data that is written to the LCD.

For each command, the input pins are latched to their desired configuration and the enable (E) signal is pulsed. This latches the internal circuitry of the LCD to whatever is connected to the input pins. The initialization procedure above sets the font, the cursor to blink at DDRAM position 0 and turns on the LCD display. All of the commands in the initialization set RS, RW as low.

The characters L-A-B are then written to the LCD. Each letter is written via 2-4 bit write operations, where RW is set to 1 (for writing).

The address of 0x05 is then sent to the address counter in 2-4 bit operations where D7 is high and RS, RW are low. In the 4-bit write operations the first 4 bits are D[4..7] and represent the MSB and the second 4-bits are D[0..3] and represent the LSB. The address is 7 bits long, and is in the order of D6-D0 (D[3..0] are the 4 bits sent in the second command).

The same write operation occurs to write G,R,O,U,P, and shift to address 0x0A, then write #, : and shift to address 0x0E to write the numbers 9, 0. The result of the above is shown below, where the correct data is displayed.



## Device 4– ECE298\_

pushbutton

## Device 5– ECE298\_

potentiometer

## Device 6– ECE298\_

led

# Part 4 – System-Level Design

Insert block diagram and description