# ECTE250

## Engineering Design and Management 2

Dr. Mohd Fareq Abd Malek, Habiba Ahmed, Eva Barbulescu, Sana Sahir, Ashna Sreejith

(Attributions: Dr Abigail Copiaco, Kiyan Afsari, Nejad Alagha)

mohamedfareqmalek@uowdubai.ac.ae , habibaahmed@uowdubai.ac.ae ,

evabarbulescu@uowdubai.ac.ae , sanasahir@uowdubai.ac.ae , ashnasreejith@uowdubai.ac.ae

# Announcement

- The workshops correspond to 6% of your total mark.
- Attendance is MANDATORY.
- Students are encouraged to participate throughout the classes. Activeness in participation also influences your total mark.

# Lab 1: Introduction to Arduino

**The Basics of Arduino:**

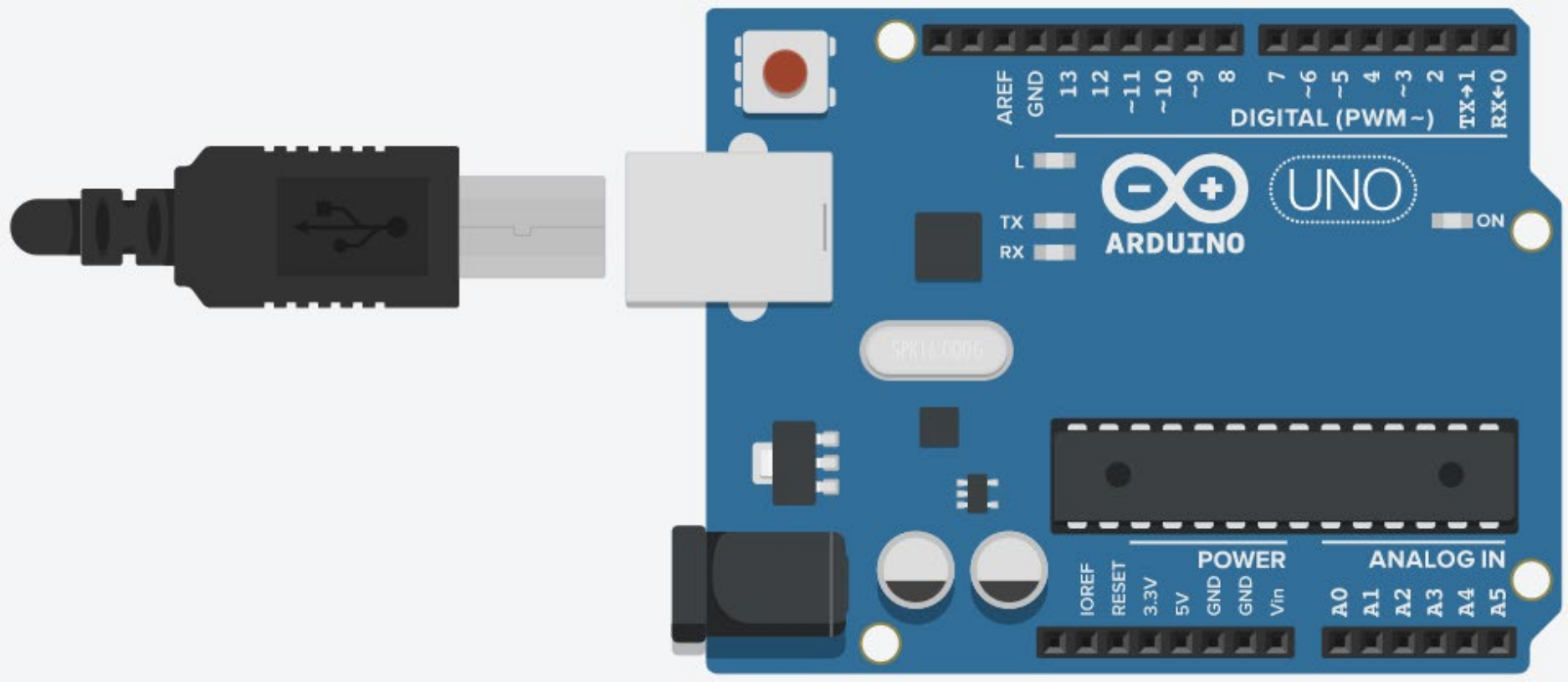
- **Recognizing your Arduino UNO**
  - The Arduino UNO will be utilized as the main microcontroller in the entire duration of the ECTE250 course. In order to upload the codes written into the board, go to *Tools -> Board, and select Arduino/Genuino Uno*.

- **Including Libraries**
  - Students may face scenarios where a new library must be incorporated to the code in order to achieve maximum efficiency. To include a new library, go to *Sketch -> Include Library*, and select the Library you would like to include from the options. New libraries can also be uploaded from a .zip folder.

- **Uploading your Code**
  - Once you are finished writing your code, click on the check button on the top left corner of the Arduino software in order to verify and check for any errors. Once the code is compiled successfully, go to *Sketch -> Upload*.

- **Opening the Serial Monitor**
  - To view readings from the serial monitor, simply click on the magnifier icon on the top right of the Arduino software.

# Arduino UNO


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

# Arduino Projects

- **Motorized Pinwheel (pp. 95-101)**
  - A motor that spins when you press the button

**Configuring your pins**

In order to declare names for the pins that you will be using in your code, use the format: `const int name = pinNumber`

For Digital pins,

eg.
```
const int greenLEDpin = 9; //green LED is connected to port 9
```

For Analog pins,

eg.
```
const int greenSensorPin = A0;
```

**What is the difference between declaring pins and declaring variables?**

- Variables are those that would change throughout the code. Hence, they are not declared with a constant data type. Similarly, modes of declared pins must be identified in the setup() function. This does not have to be done in the case of variables.

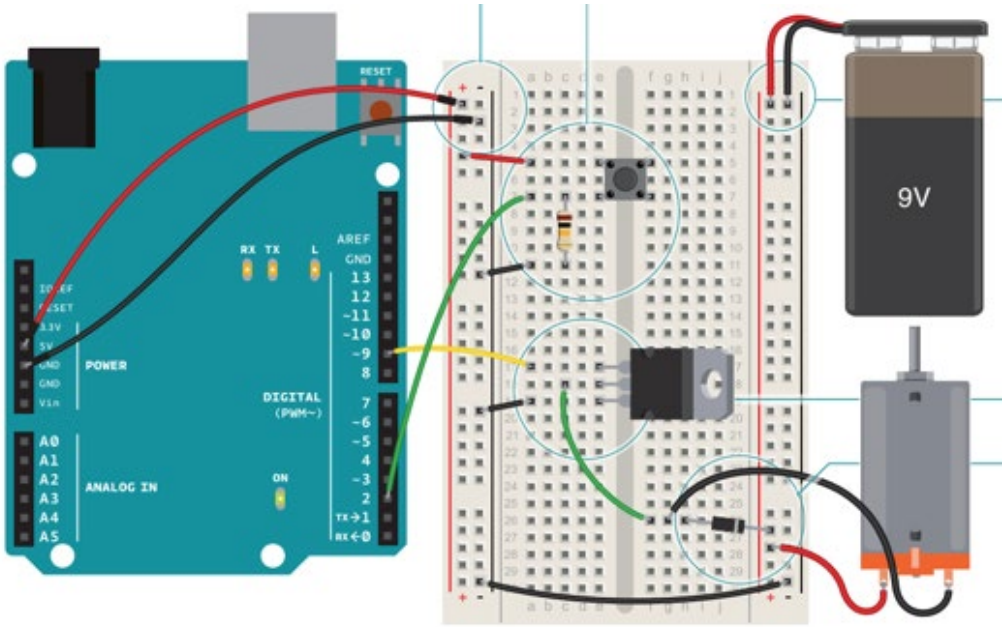
- **The setup() function**
- This is where the modes of the pins, as well as the begin function for LCD, is declared. Modes of the pins can be declared through the following format: `pinMode(pin_name, CONFIGURATION);`
  - eg. `pinMode(greenLEDpin, OUTPUT);`

## Reading and Writing to and fro Pins:

- To read from a pin, consider the code line below.
- `switchState = digitalRead(switchPin); //make sure switchState is defined`
- To write to a pin:
- `digitalWrite(motorPin, value); //value is either HIGH or LOW`

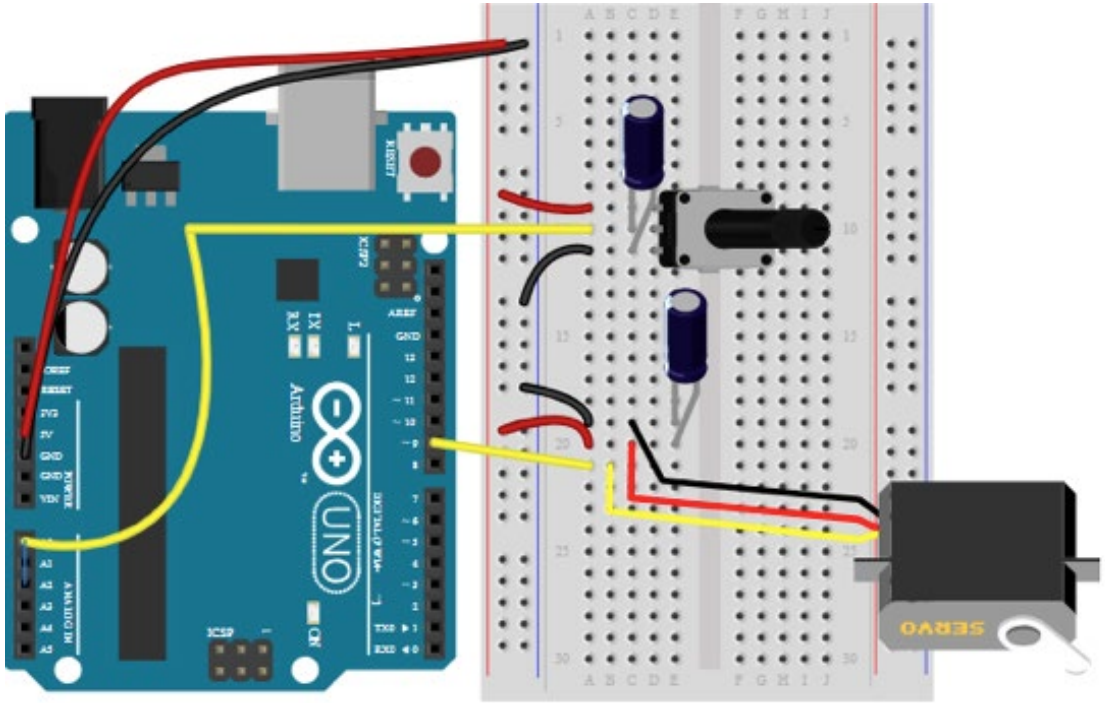
# The Connections

- **Materials Used:** IRF520N MOSFET (NMOS), 10 kohm resistor, 1N4007 Diode (Diode), Switch, DC Motor, Voltage Supply




- Follow the connections above. However, we will use a voltage supply with 9V, 1.5A setting instead of a battery.
- Upload the corresponding code onto your Arduino board: File -> Examples 10.StarterKit_BasicKit -> p09_MotorizedPinwheel.
- Why do we need a diode parallel to the motor?

- **Mood Cue (pp. 63-69)**
  - Potentiometer identifies the angle by which the servo motor spins, it also displays this in the Serial monitor
  - **Materials Used:** Potentiometer, Servo Motor, 2 100 uF capacitors



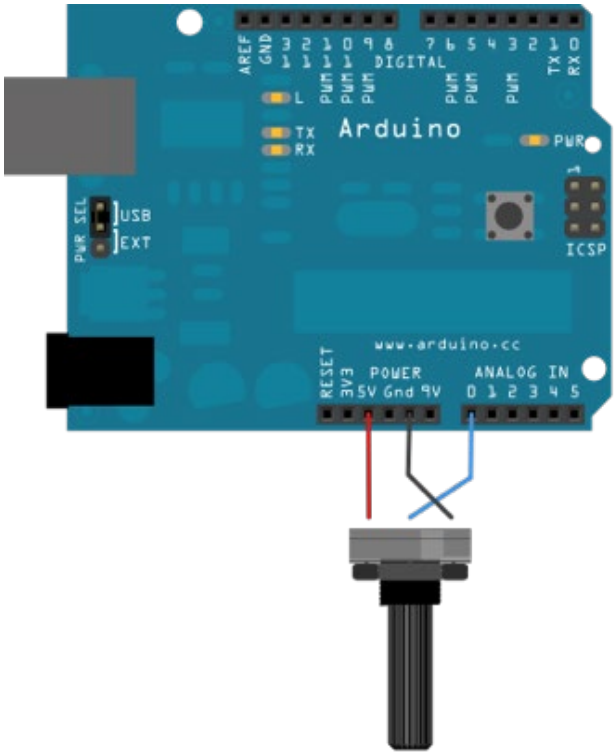

  - Upload the corresponding code onto your Arduino board: File -> Examples 10.StarterKit_BasicKit -> p05_ServoMoodIndicator

# Lab 2: ADC and LCD

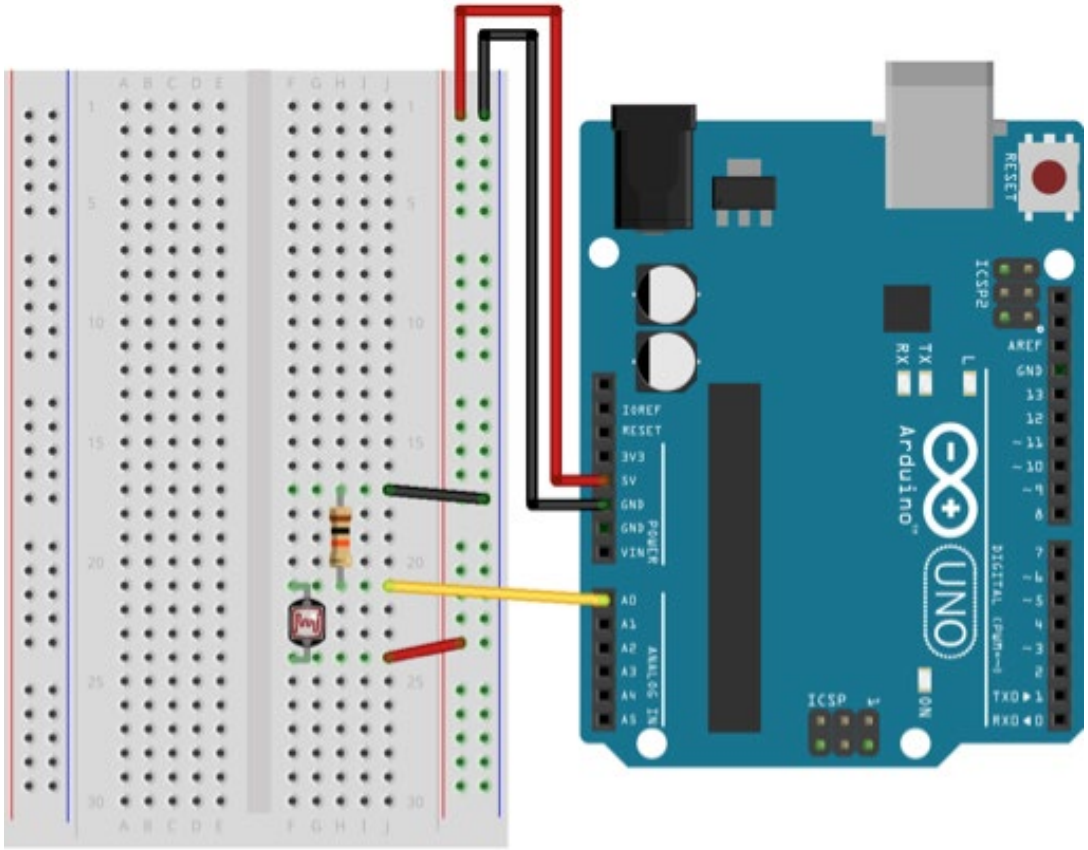
- **Analog-to-Digital Converter (ADC)**
  - A circuit that converts an analog voltage into a digital number representing that voltage. This circuit is built-in to the microcontroller, and is connected to the analog input pins A0-A5.
- **Photoresistor** - (also called a photocell, or light dependent resistor). A variable resistor that changes its resistance based on the amount of light that falls on its face.
- The analogRead() function converts the input voltage range, 0 to 5 volts, to a digital value between 0 and 1023. This is done by a circuit inside the microcontroller called an analog-to-digital converter or ADC.

# ADC Exercises

- AnalogInput
  - File -> Examples -> 0.3 Analog -> AnalogInput
  - **Controls the delay of blinking LED according to the potentiometer.**
  - Follow the following schematic (use a breadboard), add a colored LED between pin 13 and GND.



- AnalogReadSerial
  - File -> Examples -> 0.1 Basics -> AnalogReadSerial
  - Uses photoresistor (a light dependent resistor) to see the variation in the amount of voltage running on that section of the voltage divider.
  - Follow the following schematic , and observe the readings from the Serial monitor:



# Lab 2: ADC and LCD

- **Liquid Crystal Display (LCD)**
  - a tool used to display alphanumeric characters. The LCD included with your kit has 16 columns and 2 rows, for a total of 32 characters.

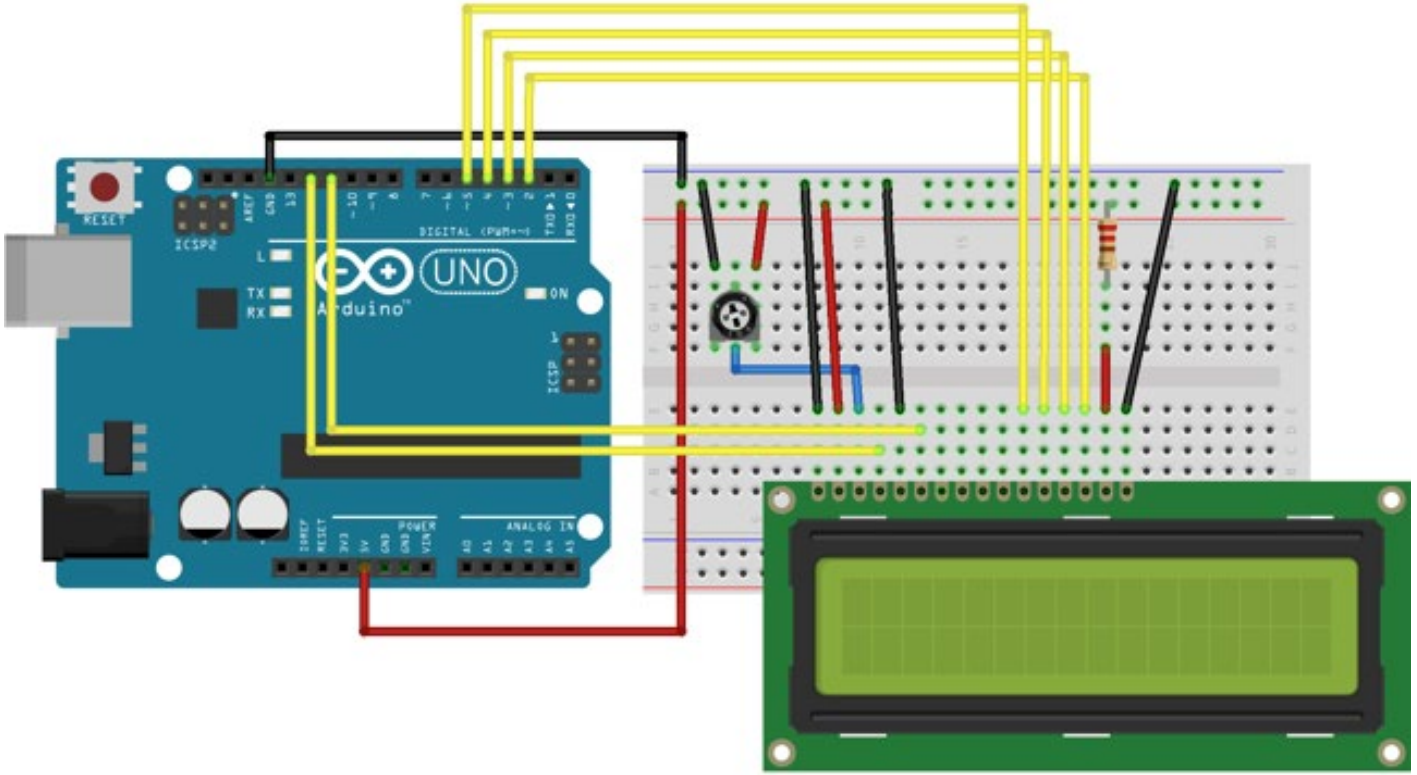
- **Using the LCD:**
  - To include the library in your code, simply type the name of the library within the brackets enclosed by the #include command. After this, the LCD must be defined based on what pins are being used to bridge the communication, as per the following example:
  - `#include<LiquidCrystal.h>;`
  - `LiquidCrystal lcd(12,11,5,4,3,2); //pins 12,11,5,4,3,2 are used`

- Initialization of the LCD is also necessary. This can be done by the `lcd.begin(16,2);` command. The reason why we used 16 and 2 as variables is because the LCD that we are working with works for 16 columns and 2 rows.

- Finally, the command `lcd.setCursor(0,1);` can be used to go to the next line.

# LCD Exercises

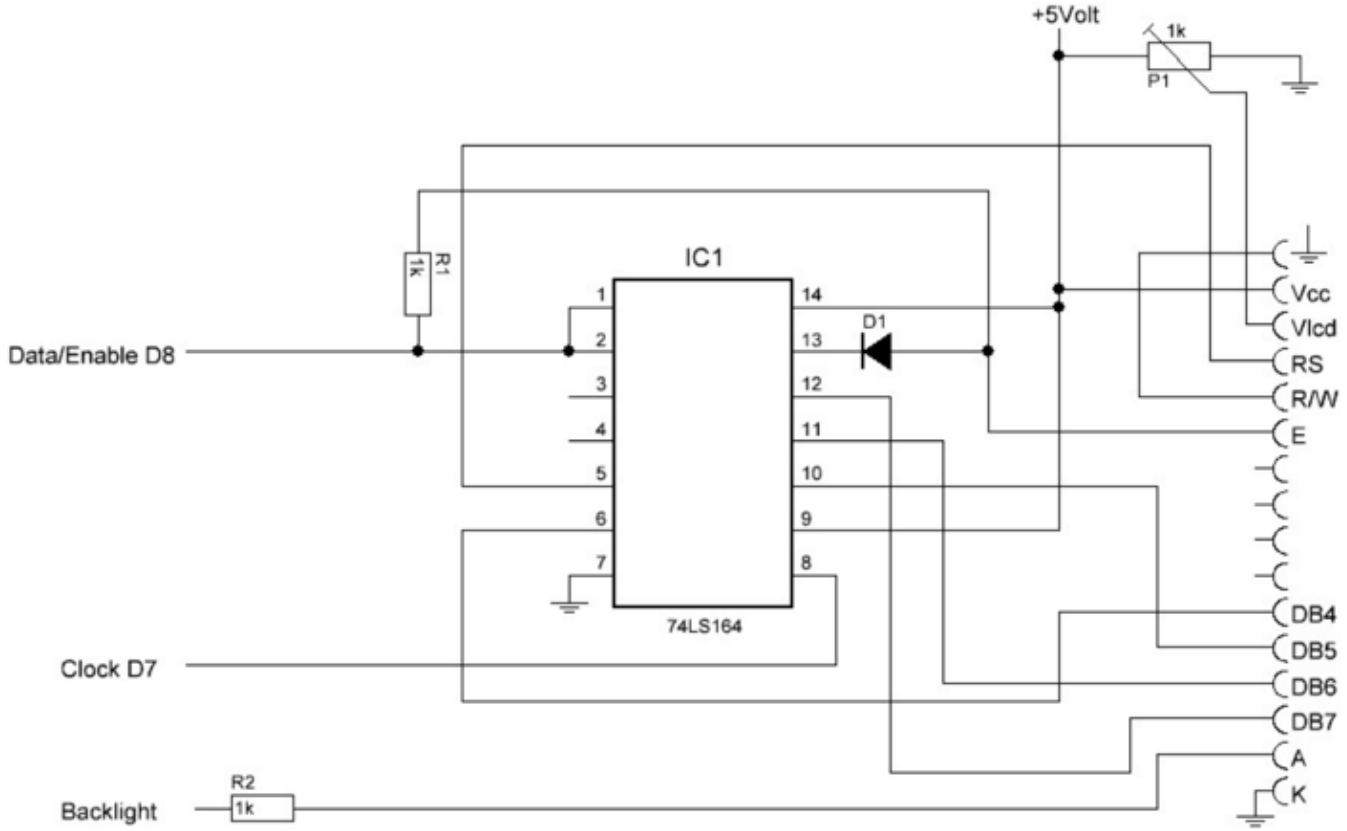
- Hello World
  - File -> Examples -> LiquidCrystal -> HelloWorld
  - Displays "Hello World" on the LCD screen. Uses the potentiometer to change the contrast of the letterings.
  - This method uses 6 pins from the Arduino



# TWI and Shift Register LCD Connections

- Materials Used: 16x2 LCD Screen, 2 pcs of 1 kohm resistors, 75HC164 Shift Register, Diode, 0.1uF decoupling capacitor

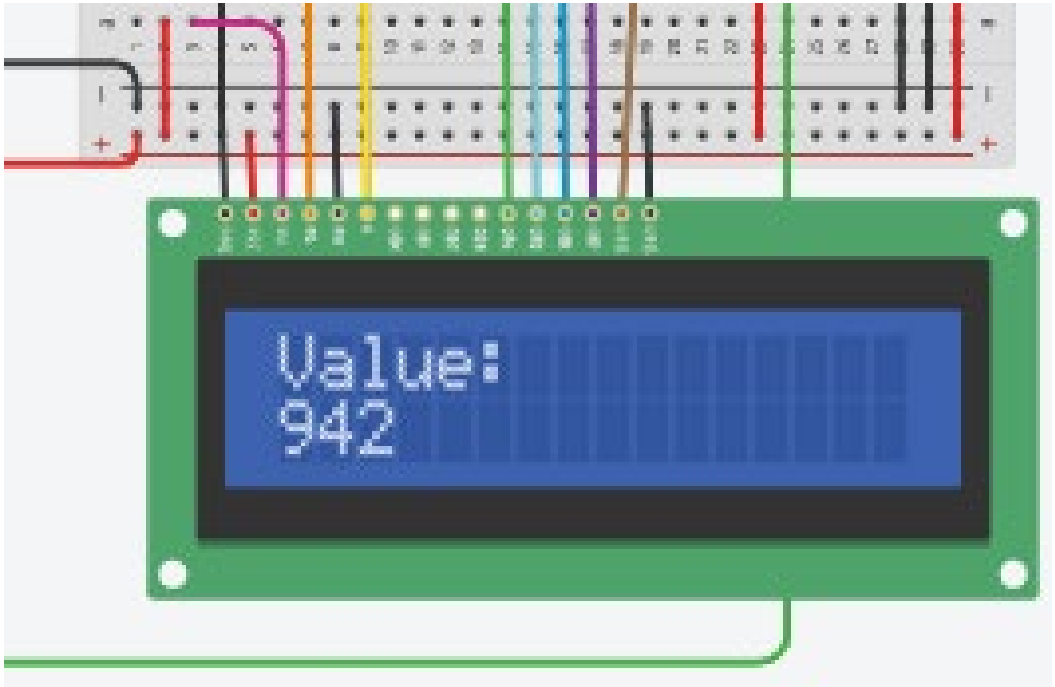
- Install the library. Follow connections below. Connect backlight to the GND, and connect a 0.1uF capacitor between the Vcc and GND of the 74HC164 shift register.



# Challenge:

Problem Statement:

- In this exercise, we will modify the functionality of the projects just examined, by combining them together!
- In exercise 2, instead of displaying the reading in the Serial Monitor, display it on the LCD!



# Lab 3: Arduino Ethernet Shield

**What is the Ethernet shield?**

- It allows you to connect your Arduino to the internet.
- A unique IP address will be given to each team, which will be utilized until the end of the semester.

**What is TCP and UDP?**

- TCP (Transmission Control Protocol) is connection oriented, whereas UDP (User Datagram Protocol) is connection-less.

## STEP 1 : Connections

1. Connect from the Router to your Ethernet Shield using an Ethernet cable
2. Place Ethernet shield on the top of your Arduino
3. Connect your Arduino to your computer via USB Port

## STEP 2 : Connecting to ECTE250 Network

1. Connect to the ECTE250 Network, password: ecte250uowd

### For Arduino:

1. Select Sketch -> Include Library -> Manage Libraries -> Select type as "Updatable"
2. Find Ethernet, select version 2.0, and install
3. Restart the Arduino IDE

### For Processing:

1. Download Processing from the link in your lab notes.
2. In Processing, go to Sketch -> Import Library -> Add Library -> Search for 'UDP' -> Install

# Section 1: Web Client and Web Server

## Key Changes in the Code:

1. Change "google.com" to "google.ae"
2. Chape IP address to your IP address

## Web Client:

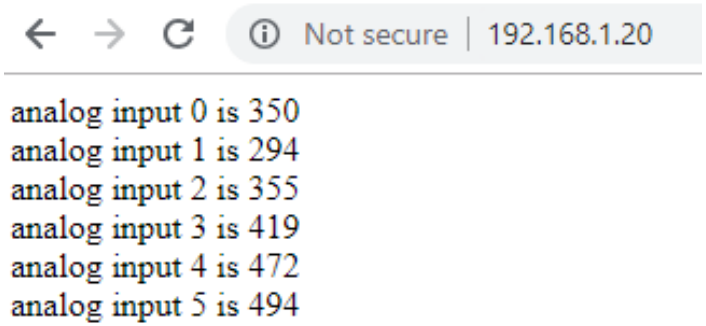
- File -> Examples ->Ethernet -> WebClient

The output should be html code after you search through the serial window.

## Web Server:

- File -> Examples ->Ethernet -> WebServer

When you type your IP address http://192.168.1.XXX to any web browser, ANALOG INPUTS 1-5 will be shown.



# Section 2: Sending and Receiving Strings via UDP

File -> Examples -> Ethernet -> UDPSendReceiveString

## Key Changes in the Arduino Code:

1. Change IP Address to the ones assigned to your team.
2. Change the localPort to 8000.

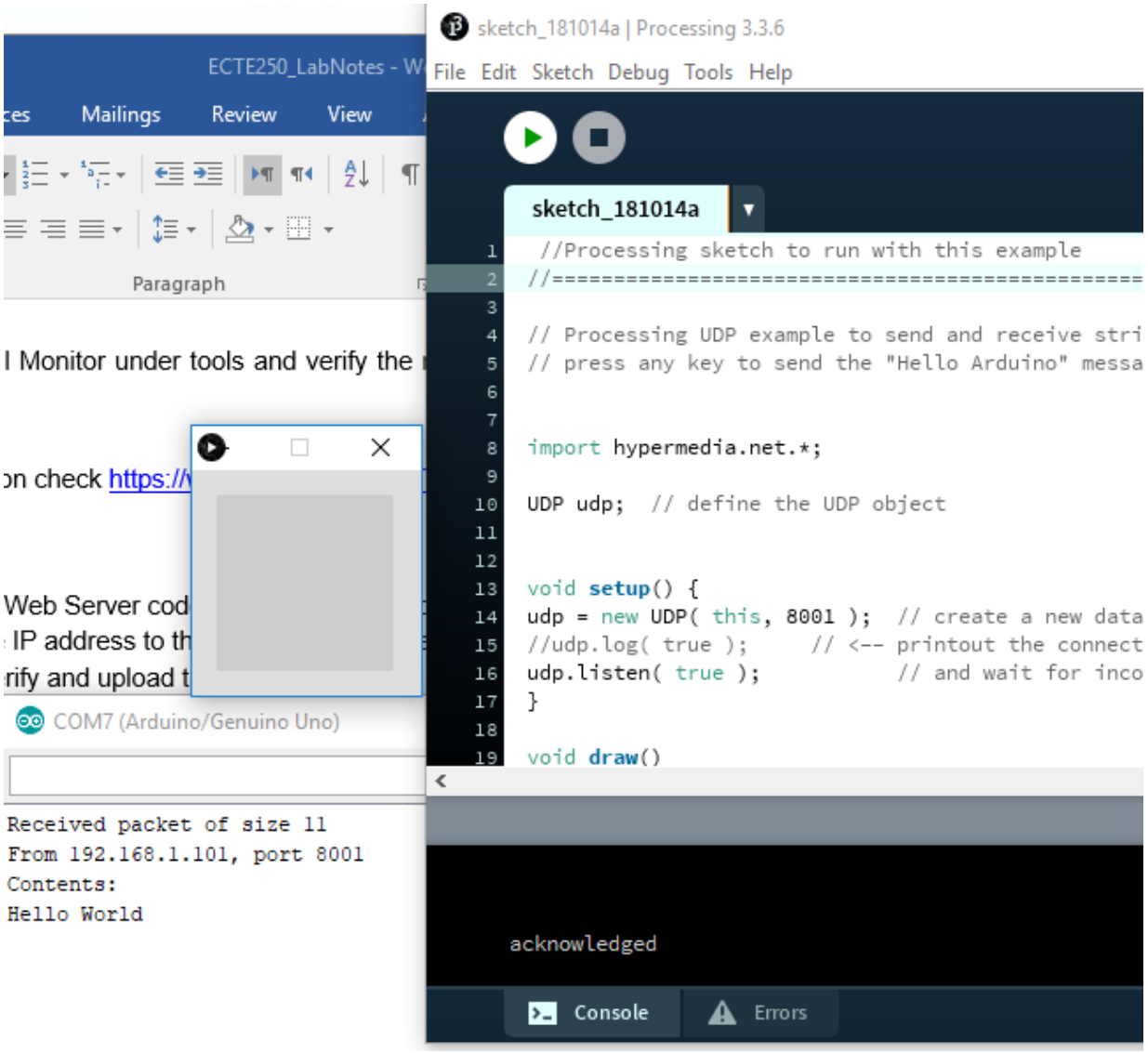
## Key Changes in the Processing Code:

1. Modify the code including the IP address of your Arduino board (change this line String ip = "192.168.1.177"; to your IP)
2. Change the new UDP to 8001, and port to 8000.

## UDPSendReceiveString:

When you click on any key while the processing small gray window is selected, "Hello World!" will be displayed in the Arduino Serial Monitor, and "Acknowledged" will be displayed in the Processing's console.

# You are receiving from port 192.168.1.101 for Processing

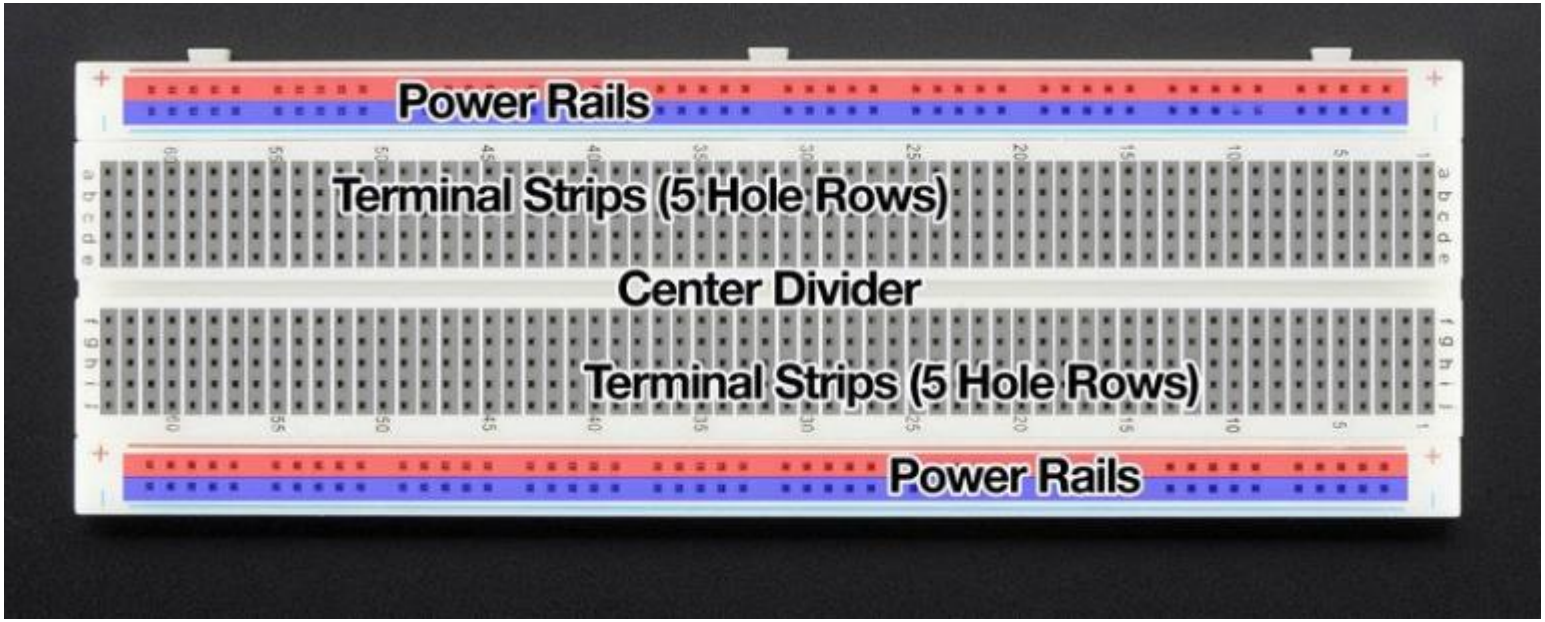


# Lab 4: Soldering

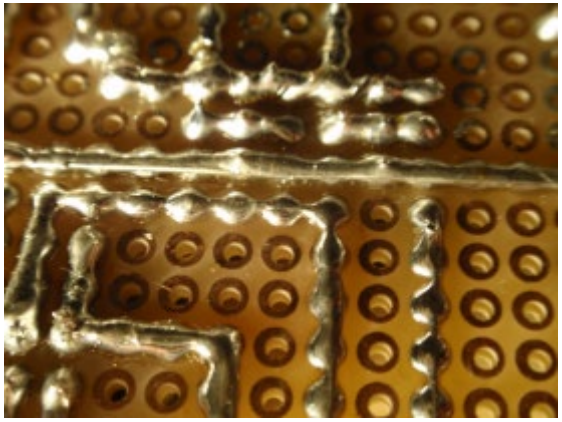
- **SOLDERING TIPS:**

1. Ensure the soldering iron is clean by dipping the tip of the iron in a slightly damped sponge.
2. Preheat the soldering iron first before using it (330 degrees Celsius)
3. Wear safety goggles to minimize risks. Remember, we are dealing with 330 degrees Celsius!
4. Always return the soldering iron to its stand when it is not used. Placing it on the counter may damage the tables and damage high voltage wires.
5. Do not touch wires that need to be soldered. Keep the soldering iron as far from your skin as possible. Clamps, tweezers, or the third hand can be used to hold two wires in place if these had to be soldered.
6. Do not inhale the soldering fumes, they are highly toxic.
7. Always turn the soldering iron off when not used.

# The Breadboard



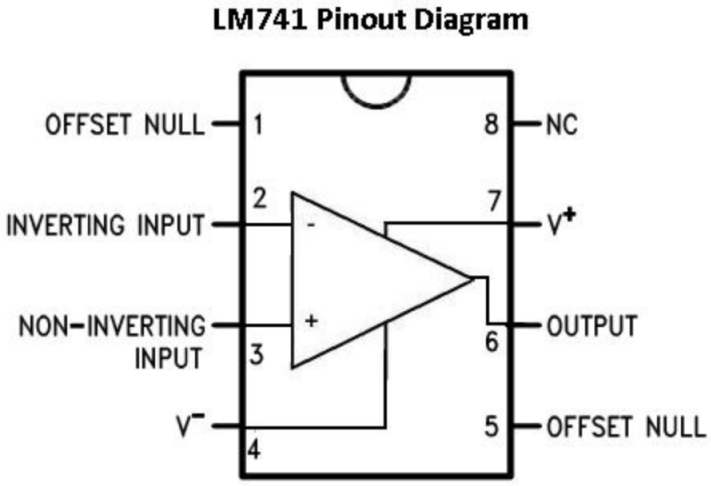
# Perfboard Prototyping

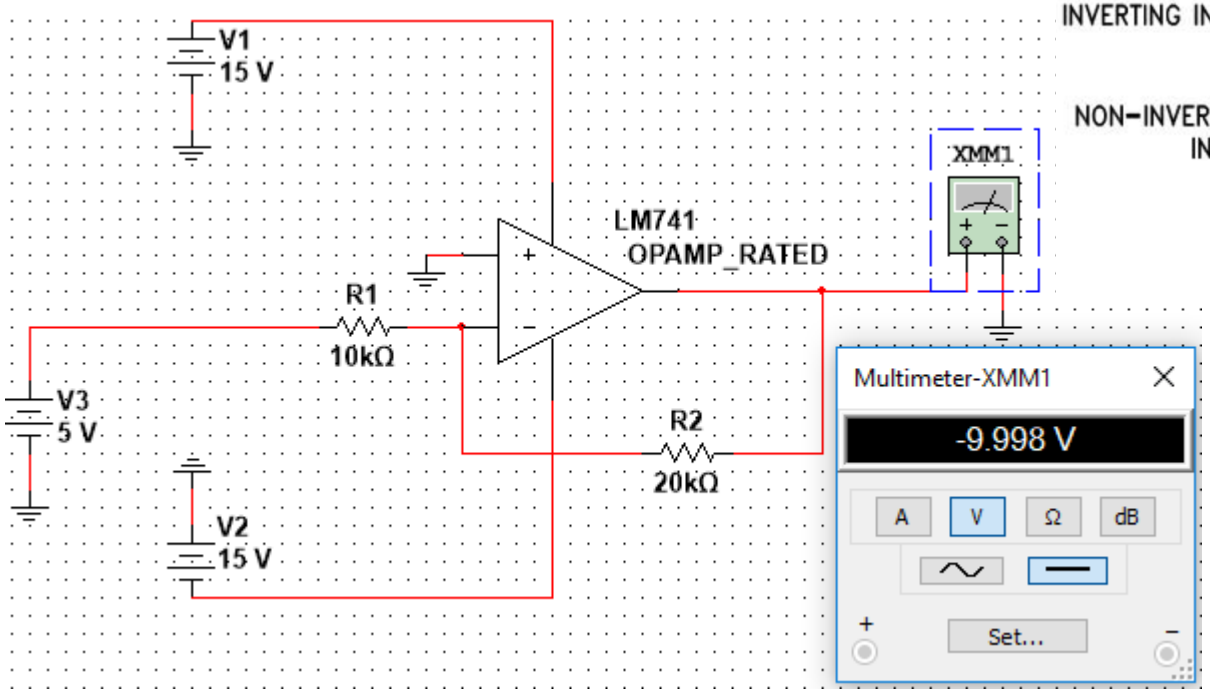


- A Perfboard gives you freedom in placing the components, as there are no lines that are connected to eachother. Hence, you will be making the connections custom-made as per your needs!

- There are 2 ways to prototype on a Perfboard:
  - Drawing the solder line
    - You can connect pins from the back of the perfboard by drawing a line through the use of solder in between the pins.
  - Connecting through wires
    - You can connect pins from the back of the perfboard by connecting the pins with a wire from the front side, and soldering the wires onto holes at the back.

# The Circuit:
# The Inverting Amplifier




An operational amplifier (op-amp) is a DC-coupled high-gain electronic voltage amplifier with a differential input and a single-ended output.

In this inverting amplifier above, the output is given as follows:

Gain = Vout/Vin = -Rf/Rin

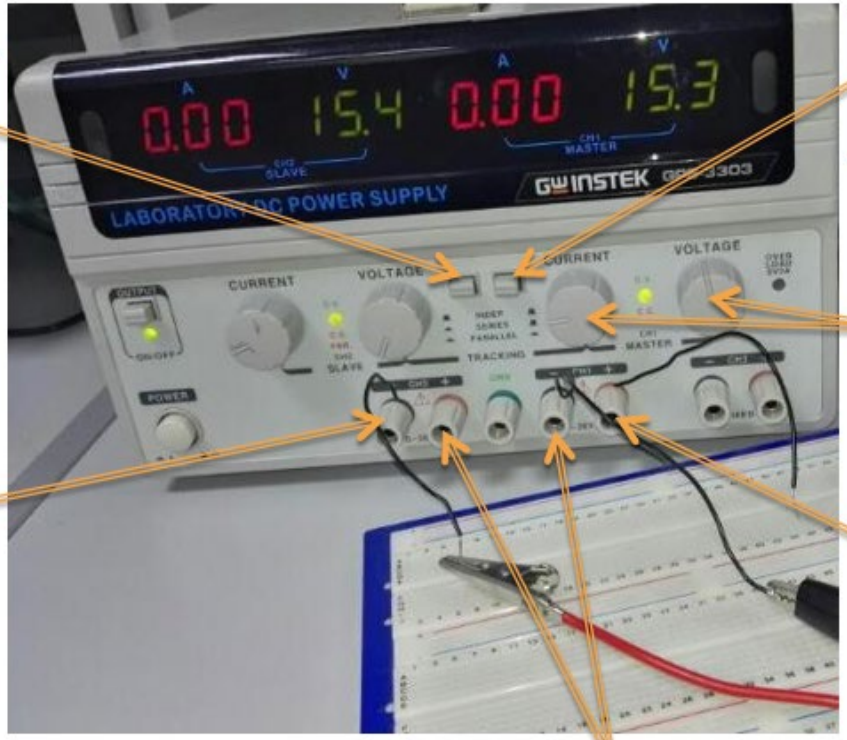
Vout = - Rf/Rin (Vin)

= - 20k/10k (5)

= - 10 V

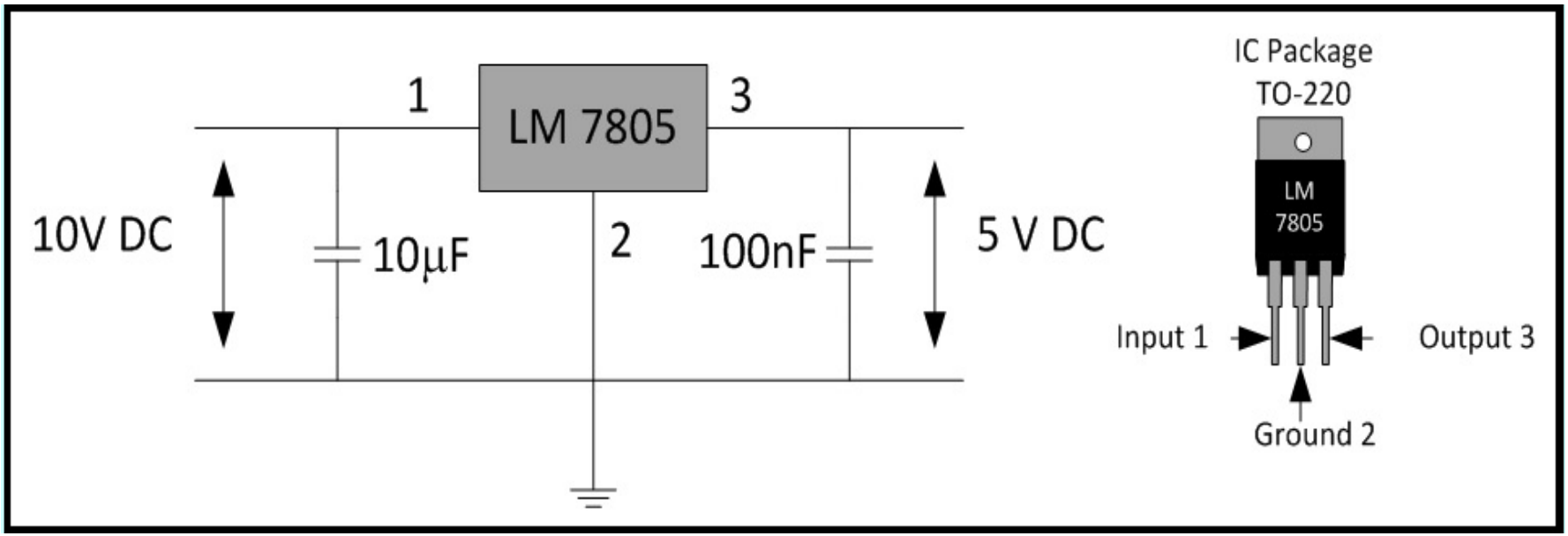
# How do we make this work?

The -15V and +15V Configuration:




The 5V DC Supply: connect it to the fixed 5V DC voltage from the voltage supply.

The LM741 IC: leave open the unused bits, while connect the rest of the pins according to the multisim circuit.


1
LM 7805
3
2
10V DC
10µF
100nF
5 V DC
IC Package
TO-220
LM
7805
Input 1
Output 3
Ground 2

# Lab 5: State Machine Design

- Boole-Deusto is a simulation software that facilitates the design and testing of state machines, as well as the generation of the translation of state machine designs into circuitry.
- Multisim is a simulation software used to design and verify the functionality of designed circuits.
- **Requirements:**
- One active-high switch (input)
- UDP messages from the Internet (input and output)
- One LED (output)
- LCD Display (output)

# Functionalities Required

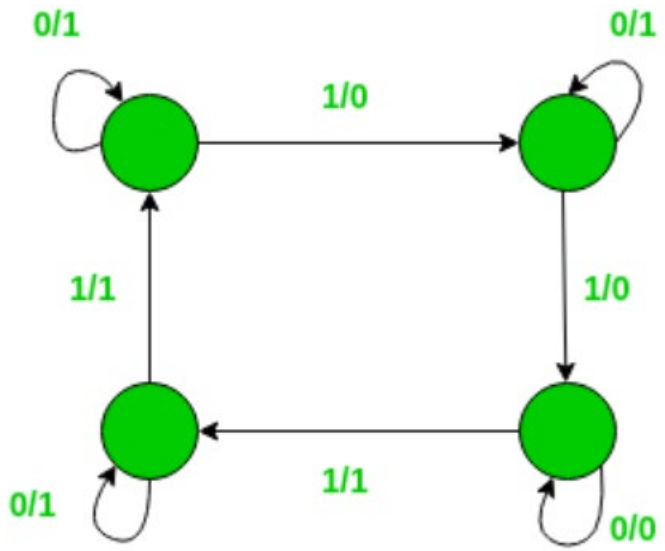
- At startup (or at reset) the LED is off and nothing is displayed on the LCD (State 00)
- When the switch (Input 1 = 1) is ON, the LED goes ON (State 01). If no message from the internet (Input 2 = 0) is received while the LED is ON, and if the switch goes to OFF (Input 1 = 0), also the LED goes OFF (going back to a state identical to the startup state – State 00).
- While the LED is ON (State 01), if the system receives the message 'YES' from the internet (Input 2 = 1), the LED stays ON, and the LCD displays 'HELLO' (State 11).
- While displaying 'HELLO' on the LCD, and the LED is ON (State 11), any change on the switch is ignored (Input 1 = X)
- While displaying 'HELLO' on the LCD, and the LED is ON, if the system receives the message 'NO' (Input 2 = 0) from the internet the display is cleared (State 01)
- When the LCD display changes from clear to 'HELLO', the message hello is also transmitted over the internet (no message has to be sent when the display changes from 'HELLO' to clear).

Outputs: LED, Message Display

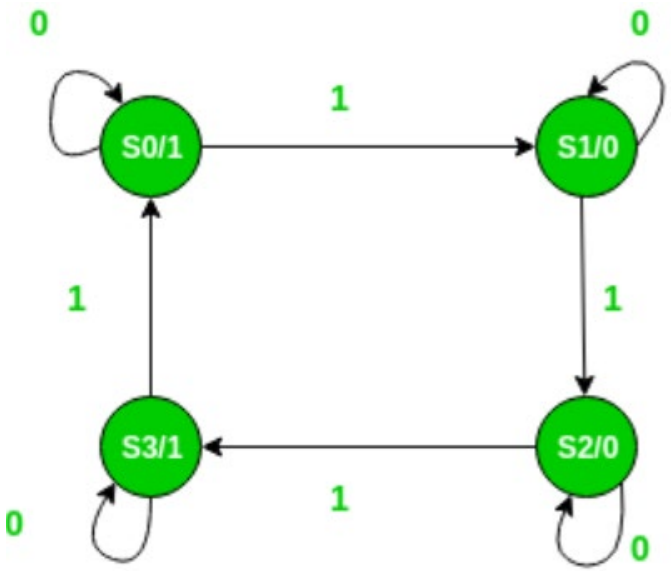
Inputs: switch, Message Input

# Types of State Machines

- **Mealy Machine**
  - defined as a machine in theory of computation whose output values are determined by both its current state and current inputs. In this machine at most one transition is possible.

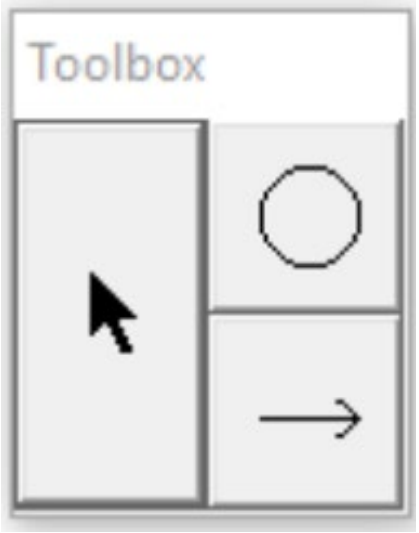



- **Moore Machine**
  - defined as a machine in theory of computation whose output values are determined only by its current state.



# How to operate Boole-Deusto?

1. Select 'Finite State Machines'.
2. A clean canvas will appear, along with the following toolbox:

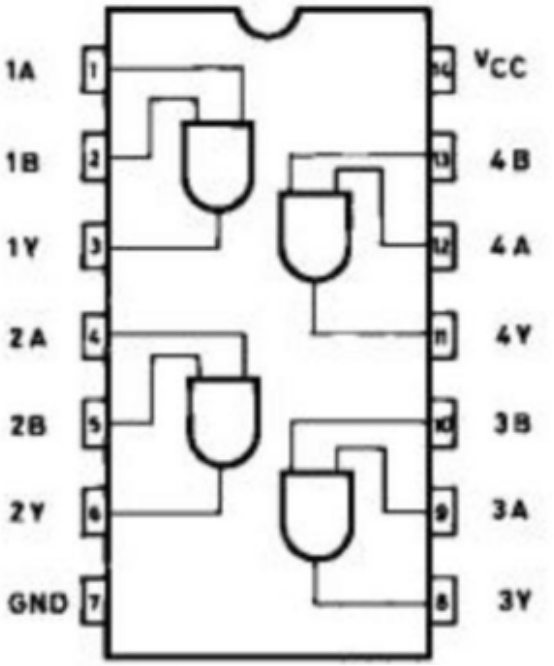


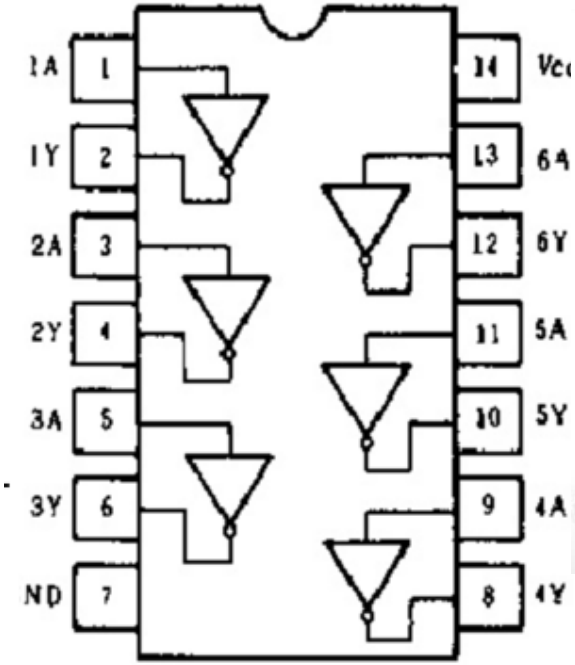

- Use the circle to create a state (just select the circle, and click on the canvas where you want the state to be placed).
- Use the line arrow to draw transitions (select the state you are coming from, then click the state you would like to go to).
- Use the cursor arrow to provide inputs and outputs (double click on the state or the transition line, and label accordingly).
- Use the cursor arrow to rearrange the position of your state machine design (just drag and drop).

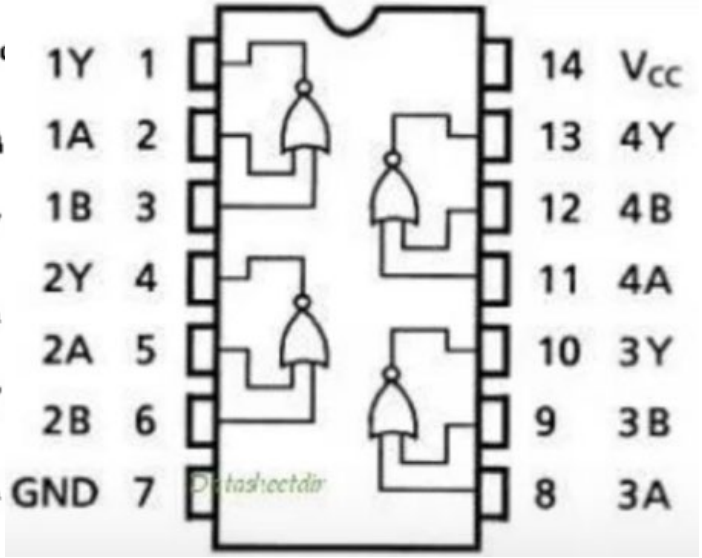
3. When ready, to simulate: Results -> Advanced Interactive Simulation
4. To view circuit equivalent: View Circuit -> With D flip-flops.

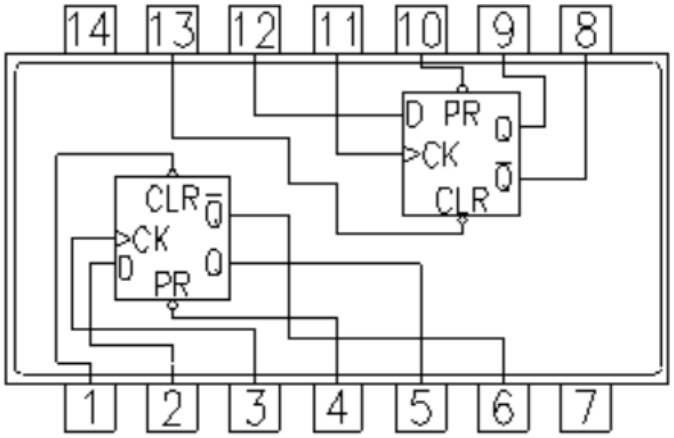
# Hardware Implementation:

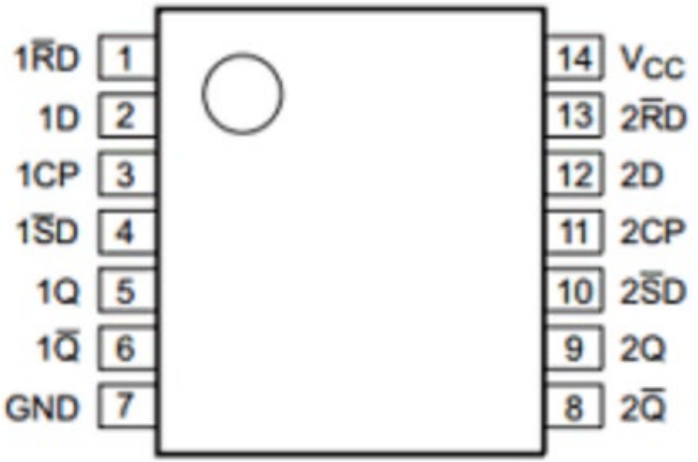
- Materials Needed: 74HC08 AND Gate, 74HC02 NOR Gate, 74HC04 NOT Gate, 74HC74 D-Flip Flop



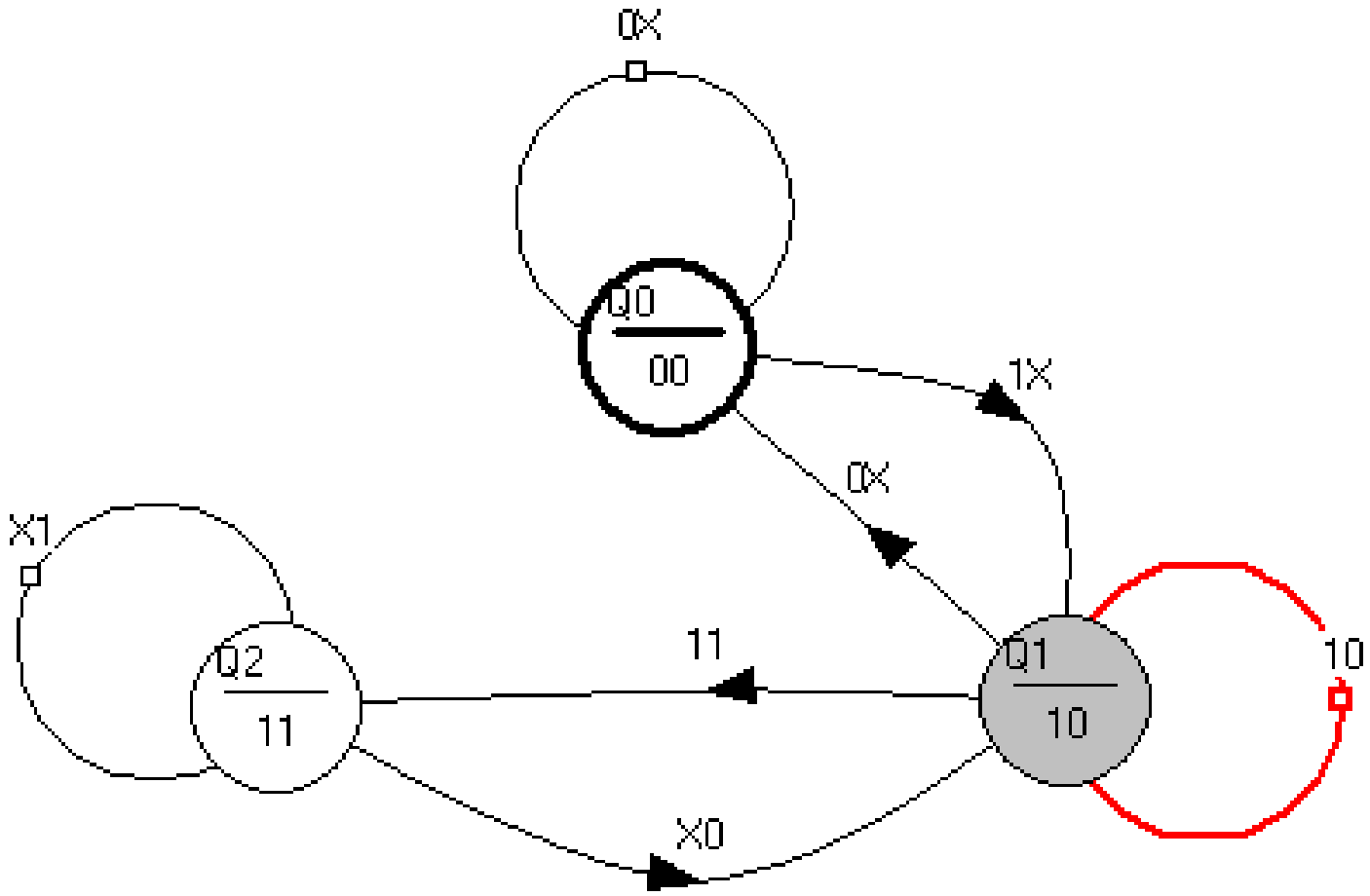



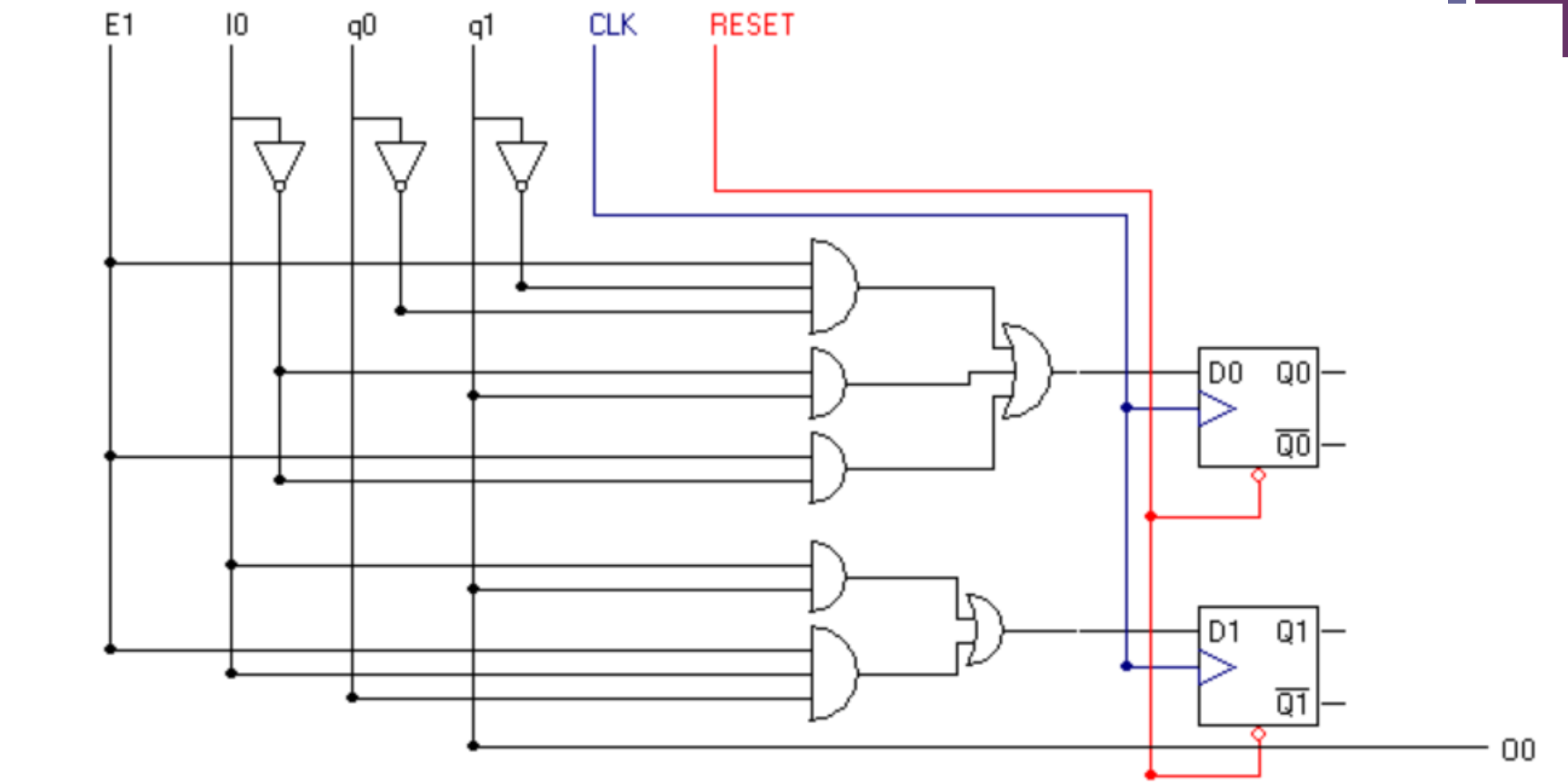



| I1 | I2 | State | |
|---|---|---|---|
| 0 | 0 | A | OFF |
| 1 | 0 | B | LED ON |
| 0 | 0 | A | OFF |
| 1 | 1 | C | ON, DISP |
| X | 0 | B | LED ON |

# How your State Diagram should look like:



# D-Flipflop Equivalent



# Making it in Multisim:

- Inputs will be represented by push buttons / switches
- Outputs will be represented by LEDs.
- Use a function generator for a clock signal (use low frequencies, square wave, with 50% duty cycle).

# Detailed Design Report Guidelines

- **4000 words maximum, 30 pages maximum.**
- **Executive Summary. (1 page)**
- **Introduction (1-2 pages).** This section introduces the product/system approved after Deliverable 1, providing a functional overview. Explain how this system/product will give you an edge over the competition (if any) and/or its impact. Explain how it aligns with the ECTE250 theme. Include a project charter in this section.
- **Design (6-10 pages).** This section includes the detailed design of the proposed product/system (*most important part and largest part of this report*). Explain how the functionalities will be implemented, include details of the design using illustrations **(flowchart, state charts, block diagrams).** The design of the state machine should be complete, related schematic and timing diagrams must be included (teams will demonstrate the state machine simulation in the laboratory session of S2-W10). Teams are also encouraged to include a final or preliminary design of the Arduino subsystem, including a flowchart of the functionality implemented in software and the source code in Appendix (teams can demonstrate this in the laboratory session of S2-W10). Use simple blocks for the components that will be designed (at schematic level) later during the Spring semester. All input and output of all blocks, including State machine and Arduino must be clearly defined.
- **Alignment (1-2 pages).** A section in which you explain how your design aligns with the requirements and constraints, and motivate what is the complexity factor of your design. Include also a list of modification from your original proposal (if any) presented in Deliverable 1, due to feedback from mentor/tutor, or proposed by the team and approved by the mentor.

- **Testing (2-3 pages)**. Brief but accurate description on the strategy you designed to test and evaluate the overall functional requirements of your product/system and those of its sub components. The testing strategy should be developed for simulation stage as well as for breadbard/perfoboard prototyping stage.
- **Plan (5 pages).** This section details the project plan, what had been achieved so far, and the forecast until project completion (i.e. complete Deliverables 3 to 8). The section (second most important part and largest part of this report) must include: Work Breakdown Structure (WBS), including work packages and their related accountable team member. A detailed Gantt chart aligned with the WBS, with the project deliverables and your internal milestones. Use the Gantt chart explain the work to be carried out until project completion, including task allocation. This section must also include the risk assessment on future activity and details on how you plan to manage those risk. For the past activities, review your performances against the original plan, indicating any delays that have occurred and their cause.
- **Budget (1-2 pages).** This section should provide the details on the budget for the prototyping parts and for labor (this includes consultation expenses). For the labor part, the report should detail how much the team has spent so far, and forecast the labor expenses until project completion (i.e. complete Deliverables 3 to 8). This section should also include a table of sale cash flow forecast detailing return of investment and profit over the next 5 years.
- **Marketing (1-2 pages).** In this brief section describe your marketing strategy for the Innovation Fair. Identify your target customers and a strategy to boost your sales.
- **References.** List of references. Use IEEE citation style. References will not be considered part of the word count. References must be cited in the text.
- **Appendix.** Include here one or more appendix including extra material. Appendices must be referenced in the text. Material included in the appendices is not considered for assessment. One appendix (not to be referenced) must include all minutes of the meetings to date (this should not be referenced in the text). Another appendix (not to be referenced) must describe the contribution of each team member with respect to the design of the system and the development of this report.

# Lab 6: Arduino Subsystem

- **Lab Session Plan**
  - Write the Code for Arduino (1.5 hours)
  - Test the Arduino and UDP Codes (0.5 hours)
- Important Examples:
  - Examples -> Digital -> Button
  - Examples -> Ethernet -> UDPSendandReceiveString
  - Examples -> StarterKit_BasicKit -> Crystal Ball
- It is advised to start with the UDPSendandReceiveString Code, and edit it according to your requirements.

# How to code

1. Definitions should have:
   - Relevant Libraries (Ethernet, EthernetUdp, LiquidCrystal)
   - LCD Pin Initializations (refer to Crystal Ball)
   - ButtonPin and ledPin initialization (refer to Button)
   - IP Addresses etc. (same as UDPSendandReceive)

2. Setup function should have:
   - Configuration for I/P for button and LED (refer to Button)
   - Lcd.begin(16,2)
   - Ethernet and Serial Initializations (same as UDPsendandReceive)

3. Loop Function should have:
   - Button State Code (for State 10 and 00) - LED on if BUTTON on, LED off if BUTTON off. (refer to Button project)
   - // if there's data available, read a packet (SAME as UDP Project)
   - Add code for UDP (if packetBuffer == "YES") / "NO"
   - Keep the remaining UDP codes

# Simulation Support Session

- State Machine
  - The heart of your project. It runs according to the specifications of your project design.
  - The design must be simulated via Boole Deusto, and with the D-flip flop circuit generated, simulate the circuit in Multisim.
- Voltage Regulator
  - Since the operational amplifiers are supplied with a +15/-15 DC power supply, voltage must be stepped down in order to fit the 5V requirement of the Arduino.
  - This circuit must be done with an LM7805 IC and capacitors. A diode can be added in order to protect the IC.
- Debouncing Circuit
  - Buttons are subject to bouncing prior to settling into a specific value. This circuit should be designed in order to stabilize the buttons in your state machine.

- 555 Timer
  - Used to demonstrate that the system is simultaneously working and generates a triangular waveform as an output.
  - Generates a waveform with the expected frequency, duty cycle, and period (calculated accordingly).

- Analog to Digital Converter (ADC)
  - Converts the output of your sensor circuit into digital values in order to maximize the use of digital pins in Arduino.
  - For example: If you are using a light sensor, values above a certain range will show an ADC output of 1000 and values below will show 0011.
  - The output voltage of the sensor circuit should be fed into the comparator of the ADC.

- Sensor Circuit
  - Multisim simulation of the sensor that you will be using for your project (eg. Flex sensor, light sensor, etc.)

- Schmitt Trigger
  - Used as an oscillator (clock circuit) to drive bits information through logic gates at a certain speed.

# Voltage Regulator

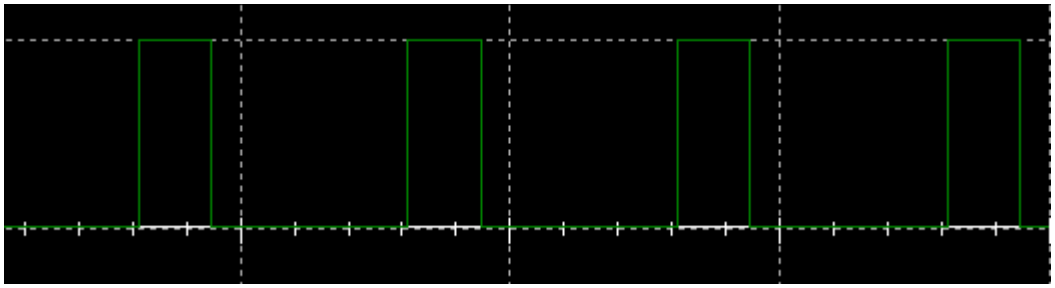
- Use an LM7805 IC (LM7805CT) coupled with capacitors, very similar to ECTE212's first lab.
- Note that you need to add one more thing to the 212's circuit!

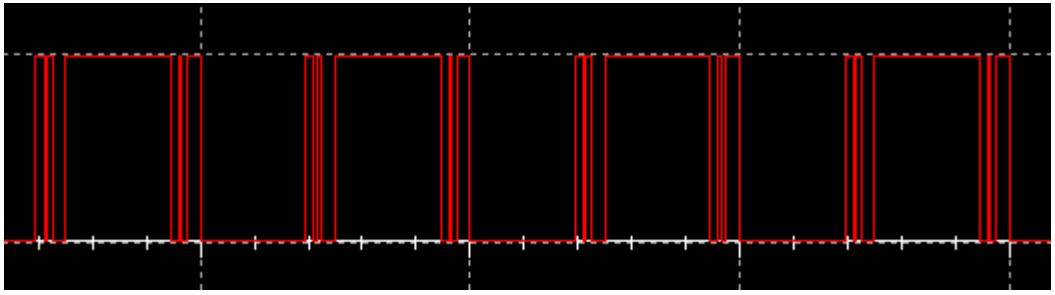
# Clock Circuit (Schmitt Trigger)

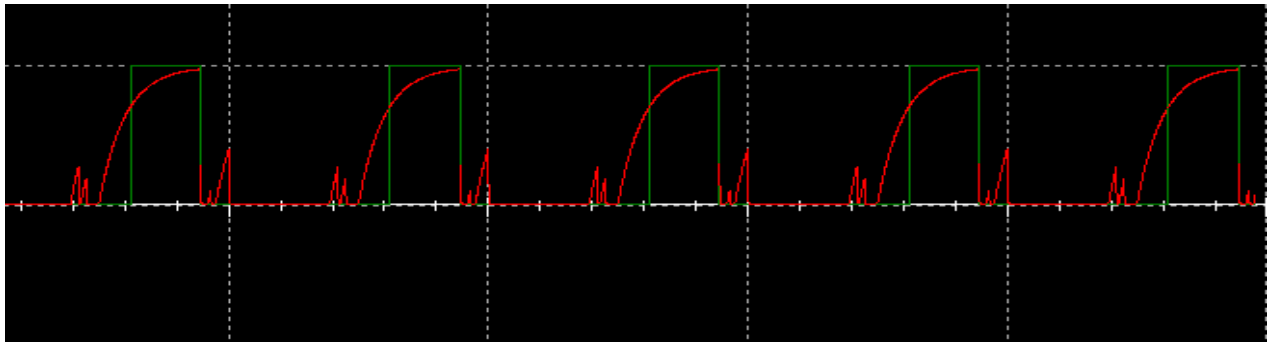
- Use a Quad 2-input NAND Schmitt Trigger (4093BD_5V), and regulate the resistor and capacitor values depending on your required frequency.
- Schmitt trigger devices are typically used in signal conditioning applications to remove noise from signals used in digital circuits, particularly mechanical contact bounce in switches.
- Schmitt trigger is a logic input circuit that **uses** hysteresis to apply positive feedback to the noninverting input of a comparator or differential amplifier. This allows the output to retain its value until the input changes sufficiently to **trigger** a change.

# Debouncing Circuit

- **TIP:** Use a Quad 2-input NAND Schmitt Trigger (4093BD_5V) for debouncing.
- Example of output graph (green is debounced, red is not):

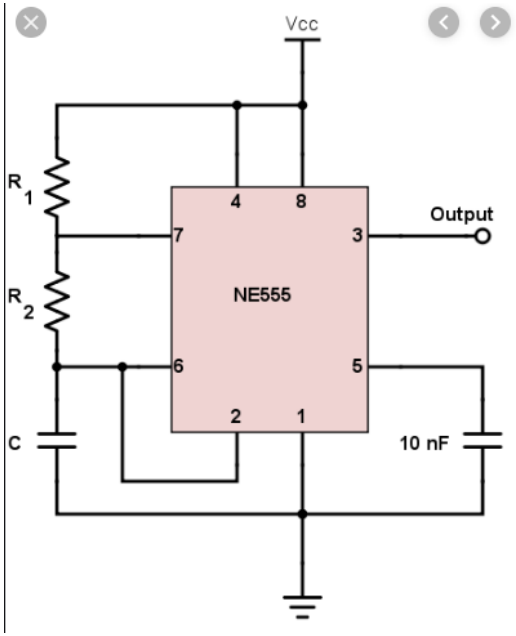


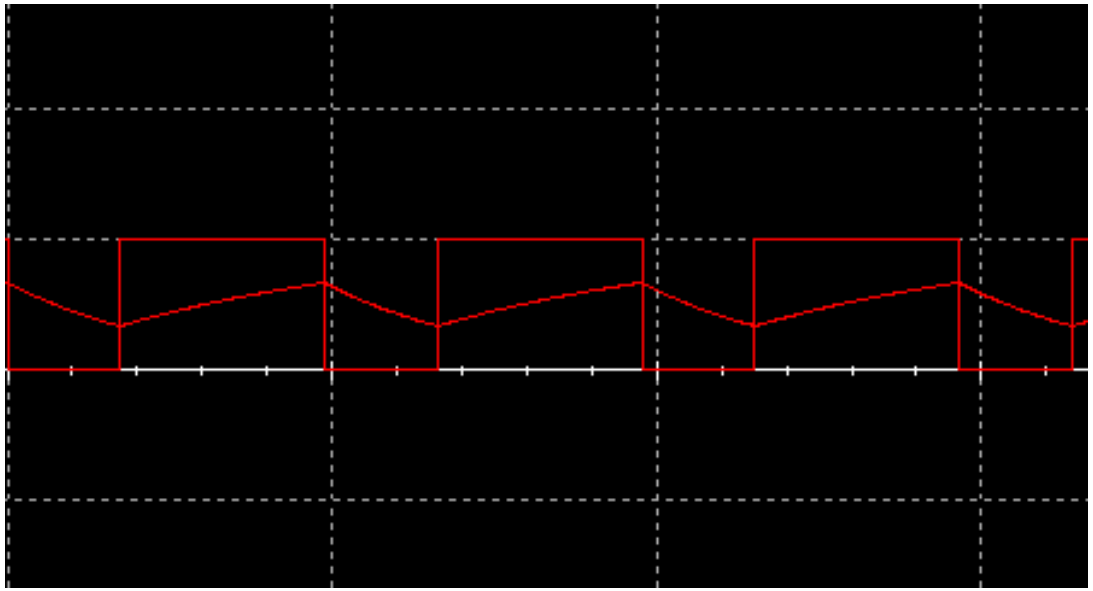




# 555 Timer (LMC555CH)

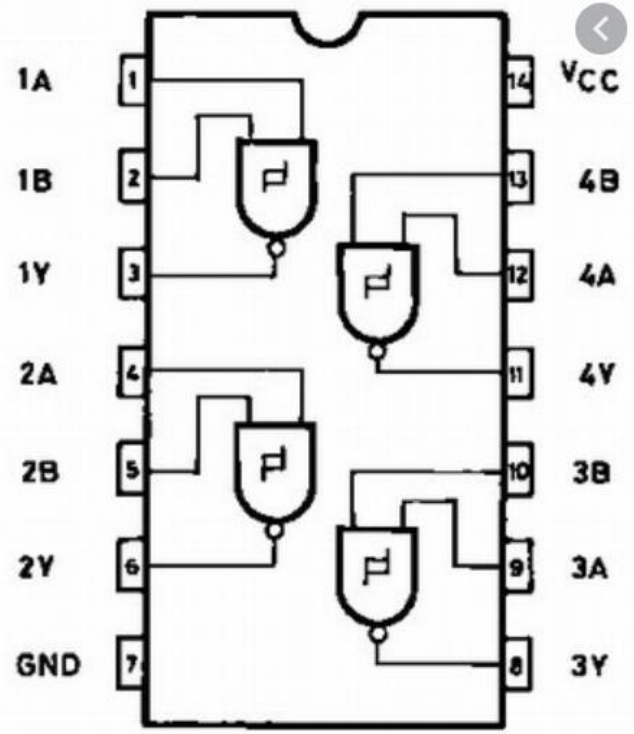
1. Use the average of the first 3 digits of your student numbers as your desired frequency.
2. Use an oscilloscope to determine the period of the circuit.
3. Duty cycle must be 60% (60% high time)
4. A digital pin from the Arduino should be used to reset the 555-timer every 100 ms.



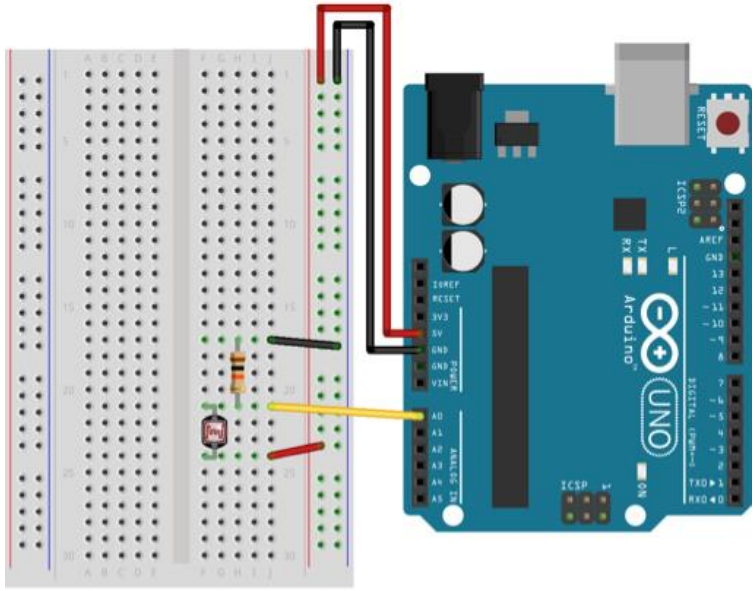
# Based on the questions I got…

- Questions about the output
  - It should simply match the output of your Multisim circuits.
- Before emailing me to check your circuits:
  - Always check your connections based on the IC pin configuration. For example, for 74HC132 (Quad NAND):



# Connection tips

- Grounds should always be connected together
- Always remember breadboard connectivity rules




- Purpose of visualization equipment
  - Oscilloscope: to visualize graphs and check AC voltage
  - Multimeter / Voltmeter: to check DC voltage

# Common Questions Received

- **How will state machines be built?**
  - Similar to Deliverable 3 – it must be made of logic gates!
  - All inputs will be replaced by SWITCHES.
  - All outputs will be replaced by LEDs.

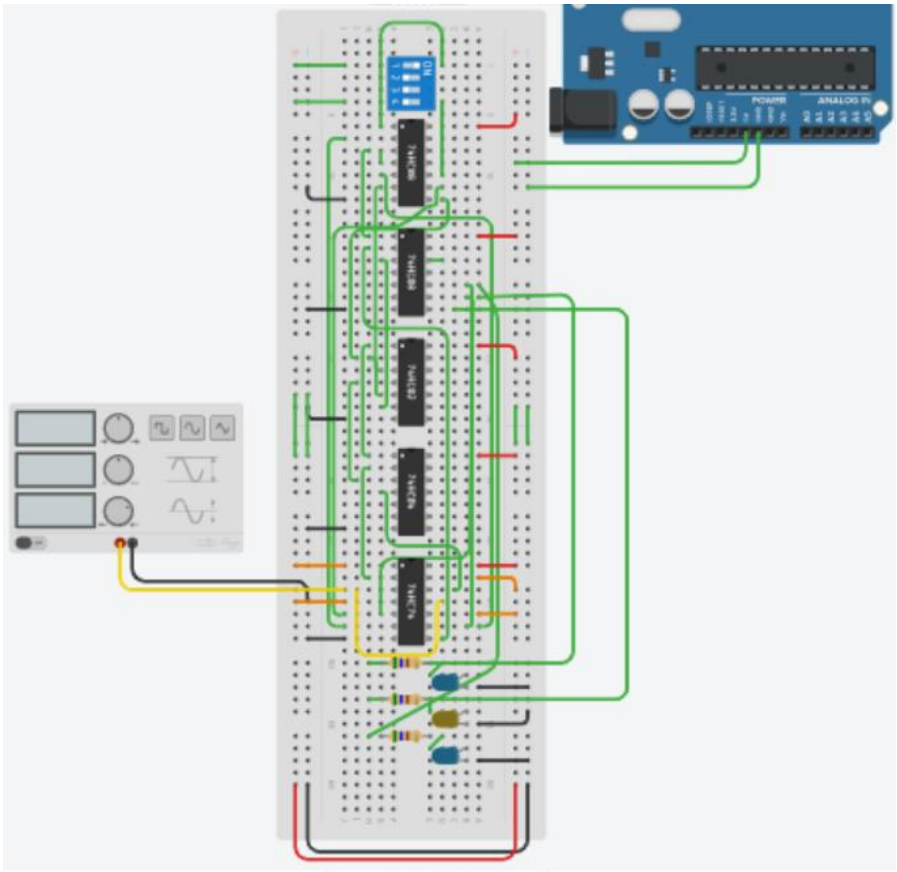


Figure 1.1: State Machine

# Testing state machine