Department of Electrical and Electronics Engineering

Project Report

Name of the Students: Mohammad Shahariar Masud Md Moyan Uddin	**ID of the Students**: 222901008 222901001
Course Title: Microprocessor and Interfacing Lab	**Course Code**: EEE 318
Trimester/Semester: Fall 2024-25	Section: 221-D3
Project Title: Designing a Digital Clock with Arduino uno	

Project Report Evaluation [25]				
COs	**Evaluation Criteria**	**POs**	**Marks Assigned**	**Marks Obtained**
CO1	Methodology and Design procedure are properly adopted.	PO3	4	
CO2	Results are properly analyzed and verified.	PO4	4	
CO3	Appropriate hardware and/or software tools are used.	PO5	4	
CO4	Assessed societal, health, safety, legal and cultural issues involved with the mini project	PO6	2	
CO5	Realized the impact of societal, environmental and sustainable development issues for the design solution of mini project.	PO7	2	
CO6	Applied professional ethics and responsibilities in the implementation of mini project.	PO8	2	
CO8	Reported and addressed knowledge, data, information, results properly	PO10	2	
CO9	Applied engineering project management knowledge and skill	PO11	2	
CO10	Applied knowledge, data and information from various multidisciplinary sources to analyze, design and implement the mini project.	PO12	3	
		Total Marks	**25**	
Name & Designation of the Course Teacher: **Mahmudur Rahman, Lecturer, Green University of Bangladesh**			**Signature & Date:**	

Name of Project: Digital Clock with Arduino Uno

Objectives: The object of this project is to design a Digital clock that displays the time in hours, minutes, and seconds.

Introduction: A digital clock is a type of clock that displays the time digitally (i.e. in numerals or other symbols), as opposed to an analog clock, where the time is indicated by the positions of rotating hands. The times derived by analog clocks come from either a pendulum or a spring. Pendulums are unusable on moving platforms, such as a ship, and springs unwind slowly as they release stored tension. The use of sweep hands allowed these mechanical time bases to be presented in a mechanically driven display. With the perfecting of multivibrator chips, electrical circuits could be built that could accurately keep time under a wide range of conditions. As the time base had switched from mechanical to electrical, the time display had to follow suit. Display devices called 7 segment displays were designed to allow the time to be shown numerically.

Required Equipment:

1. Arduino Uno R3 Board.
2. Push button- 4 pcs.
3. 7 Segment Display Common Cathode- 6 pcs.
4. Proteus software
5. Shift register Ic-4094
6. Connecting wire
7. Ground pin

Component Description:

1. **Arduino Uno R3 Board**:
   Arduino UNO R3 is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

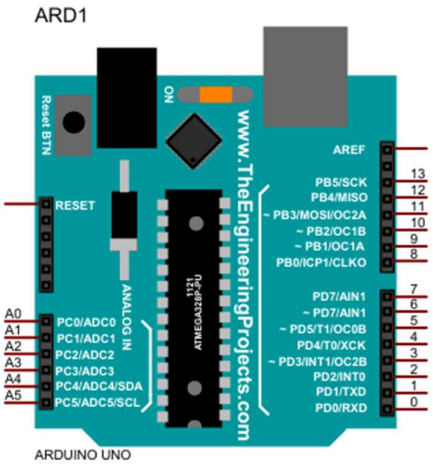

Figure 1 Arduino

2. Push Button:

A push button switch controls an action in a machine or other type of process. They are common features within the home and workplace and are also referred to as pushbutton switches or push switches. In this project push buttons are used for set time.

Figure 2 Push Button

3. Shift Resistor

The 4094 is an 8-bit serial-in, parallel-out (SIPO) shift register with a storage latch. It accepts serial data, shifts it on the rising edge of the clock signal, and outputs it in parallel or via a serial output for cascading multiple ICs. The strobe pin controls data transfer to the output latch, and the enable pin controls the output state. It operates at 3V–15V and is commonly used for expanding microcontroller I/O, driving LEDs or displays, and data communication

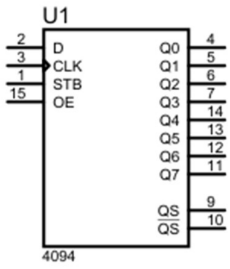

Figure3: Shift Resistor(4094)

4. Seven Segment Display:

The 7-segment displays are just seven LEDs lined up in a particular pattern. In this case, the number '8' shape we are all familiar with. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed. An additional 8th LED is sometimes used for an indication of a decimal point.

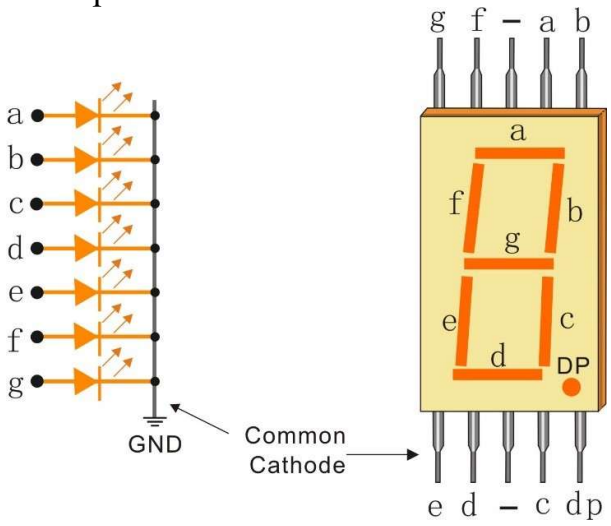

Figure 4 Seven Segment Display

Each one of the seven LEDs in the display is given a positional segment with one of its connection pins being brought straight out of the rectangular plastic package. These individual LED pins are labeled from a through to g representing each individual LED. The other LED pins are connected and wired to form a common pin.

To turn on and off a particular part of the display, you set the appropriate pin HIGH or LOW just like you would with a regular LED. So that some segments will be light, and others will be dark allowing the desired character pattern of the number to be generated on the display. This then allows us to display each

of the ten decimal digits 0 through to 9 on the same 7-segment display.

7 Segment Display Pinout:

The pinout of 7-segment displays are follows:

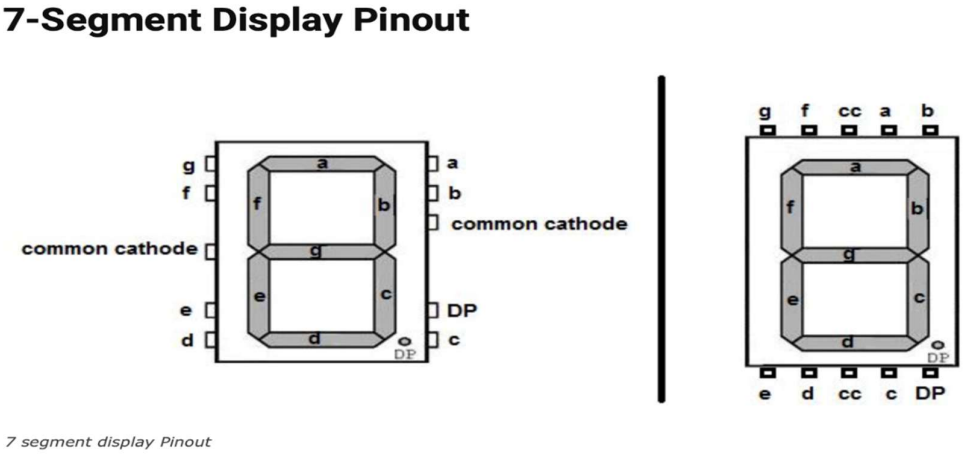

Figure5:Seven Segment Display

a-g & DP Out of 10, the 8 pins i.e. a, b, c, d, e, f, g and DP segment (decimal point) are connected to digital pins of Arduino. By controlling each LED on the segment connected, numbers can be displayed.

COM The pin 3 and 8 are internally connected to form a common pin. This pin should be connected to GND (common cathode) or 5V (common anode) depending upon the type of display.

Truth Table For 7 segment display:

Input							Output
a	**b**	**c**	**d**	**e**	**f**	**g**	
1	1	1	1	1	1	0	0
0	1	1	0	0	0	0	1
1	1	0	1	1	0	1	2
1	1	1	1	0	0	1	3
0	1	1	0	0	1	1	4
1	0	1	1	0	1	1	5
1	0	1	1	1	1	1	6
1	1	1	0	0	0	0	7
1	1	1	1	1	1	1	8
1	1	1	1	0	0	1	9
1	1	1	0	0	1	1	A
0	0	1	1	1	1	1	B
1	0	0	1	1	1	0	C
0	1	1	1	1	0	1	D
1	0	0	1	1	1	1	E
1	0	0	0	1	1	1	F

Circuit Diagram:

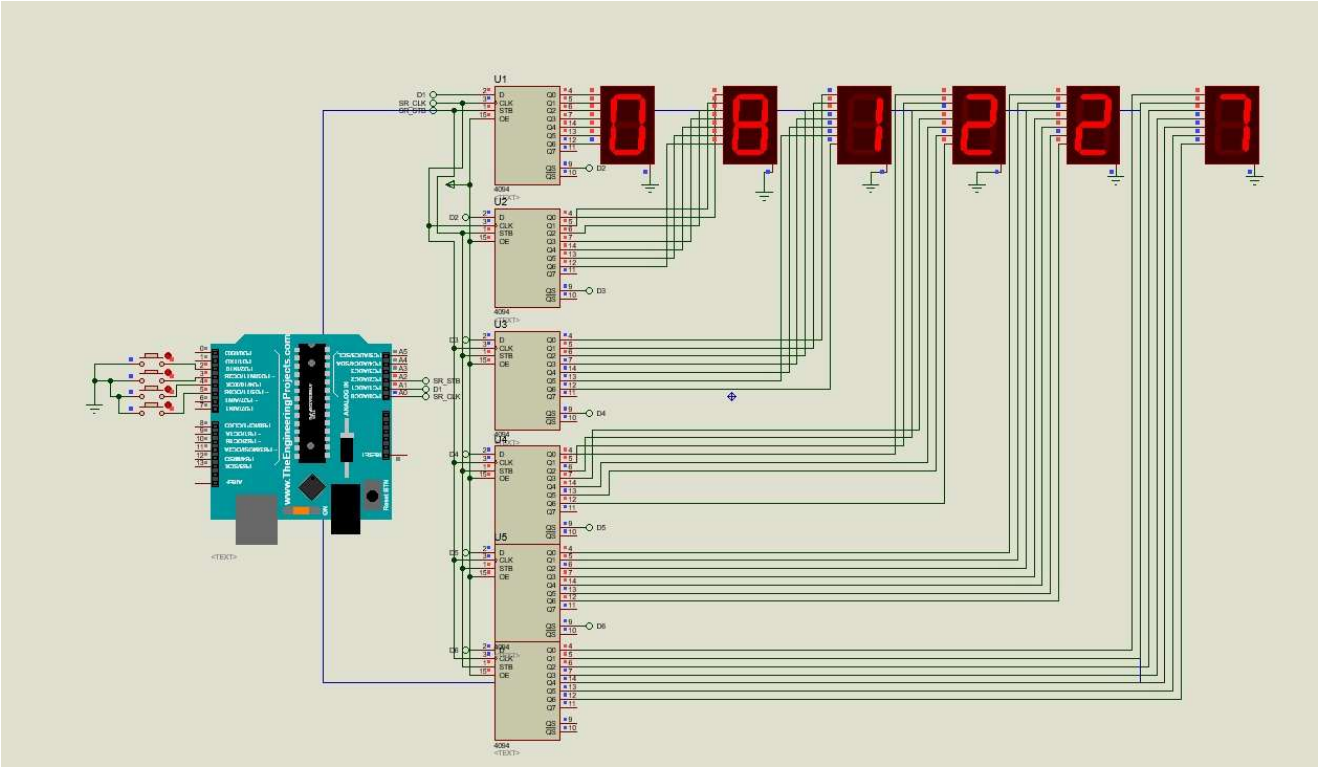

Figure 6 Circuit Diagram

Circuit Description:

1. Power Supply: Provides Supply to Arduino
2. 7 Segment Display: 7 Segment Display is the output device. In 7 Segment Display, we can see our output result
3. Arduino Uno R3: Arduino is the controller of the full circuit, and it can give the output to the 7 Segment Display.
4. Shift Register: We also used shift register to pass the data, which are connected to the 6* 7-segments display parallelly
5. Push Buttons: This is used to set Time.

Procedure:

1. First, place four 7 Segment Display in the breadboard. Then we short the G, F, A, B, E, D, C, and DP port of the 4 seven segment display together.

2. Then we connect the VCC with a resistor
3. After completing the internal connection of seven segment display, we connect the pin with Arduino.

Here is the detailed connection between Arduino, Shift Resistor and 7 segment display

Arduino	Shift Resistor
A0	CLK
A1	D
A2	STB
Shift Resistor	**7-Segment Display**
Q0	A
Q1	B
Q2	C
Q3	D
Q4	E
Q5	F
Q6	G
QS	D(Shift Resistor)

4. After that, we place 4 push buttons to control the time. And connect them with Arduino's D2 ,D3,D4,D5 pins.
5. Then we connect the Arduino with our Laptop and insert the code by using IDE software. Use used C programming for the coding.
6. Then finally we ran our circuit.

Arduino Code:

```
#define NUM_OF_DISPLAY 6

// Pin connections
int strobePin = A2;
int dataPin = A1;
int clockPin = A0;

// Switch pins
int hourSwitchPin = 2;    // Adjust hours
int minuteSwitchPin = 3;  // Adjust minutes
int secondSwitchPin = 4;  // Adjust seconds
int resetSwitchPin = 5;   // Reset all displays to zero

// Buffer for the digits to display
char shiftOutBuffer[NUM_OF_DISPLAY] = {0};

// Segment character map for digits (0-9) and blank
byte segChar[] = {
  0b00111111, // 0
  0b00000110, // 1
  0b01011011, // 2
  0b01001111, // 3
  0b01100110, // 4
  0b01101101, // 5
  0b01111101, // 6
  0b00000111, // 7
  0b01111111, // 8
  0b01101111, // 9
  0b00000000  // Blank
};

unsigned long lastUpdate = 0;          // Time tracking for display update
unsigned long lastSwitchPress[4] = {0, 0, 0, 0}; // Debounce tracking for switches
unsigned int hours = 23, minutes = 59, seconds = 30; // Starting time: 23:59:30

void update_display() {
  digitalWrite(strobePin, LOW);
  for (int i = NUM_OF_DISPLAY - 1; i >= 0; i--) {
    shiftOut(dataPin, clockPin, MSBFIRST, segChar[shiftOutBuffer[i]]);
  }
  digitalWrite(strobePin, HIGH);
}

void reset_digits() {
```

```
  hours = 0;
  minutes = 0;
  seconds = 0;
  for (int i = 0; i < NUM_OF_DISPLAY; i++) {
    shiftOutBuffer[i] = 0;
  }
  update_display();
}

void setup() {
  pinMode(strobePin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);

  // Configure switch pins with pull-up resistors
  pinMode(hourSwitchPin, INPUT_PULLUP);
  pinMode(minuteSwitchPin, INPUT_PULLUP);
  pinMode(secondSwitchPin, INPUT_PULLUP);
  pinMode(resetSwitchPin, INPUT_PULLUP); // Configure reset pin with pull-up

  reset_digits(); // Clear the display
}

void loop() {
  // Handle button presses with debounce logic
  if (digitalRead(hourSwitchPin) == LOW && millis() - lastSwitchPress[0] > 200)
{
    lastSwitchPress[0] = millis();

    // Increment hour (wrap around at 24)
    hours = (hours + 1) % 24;

    // Recalculate seconds based on updated hours
    seconds = (hours * 3600) + (minutes * 60) + seconds % 60;
  }

  if (digitalRead(minuteSwitchPin) == LOW && millis() - lastSwitchPress[1] >
200) {
    lastSwitchPress[1] = millis();

    // Increment minutes (wrap around at 60)
    minutes = (minutes + 1) % 60;

    // Recalculate seconds based on updated minutes
    seconds = (hours * 3600) + (minutes * 60) + seconds % 60;
  }
```

```
  if (digitalRead(secondSwitchPin) == LOW && millis() - lastSwitchPress[2] >
200) {
    lastSwitchPress[2] = millis();

    // Increment seconds (wrap around at 60)
    seconds = (seconds + 1) % 60;

    // Recalculate the full time after seconds increment
    seconds = (hours * 3600) + (minutes * 60) + seconds;
  }

  if (digitalRead(resetSwitchPin) == LOW && millis() - lastSwitchPress[3] > 200)
{
    lastSwitchPress[3] = millis();
    reset_digits(); // Reset time and clear display
  }
  // Update time every second
  if (millis() - lastUpdate >= 1000) {
    lastUpdate = millis();

    // Increment the second and wrap around at 60
    seconds = (seconds + 1) % 60;

    if (seconds == 0) {
      minutes = (minutes + 1) % 60;
      if (minutes == 0) {
        hours = (hours + 1) % 24;
      }
    }
    // Update the display buffer
    shiftOutBuffer[0] = hours / 10;    // Tens place of hours
    shiftOutBuffer[1] = hours % 10;    // Units place of hours
    shiftOutBuffer[2] = minutes / 10; // Tens place of minutes
    shiftOutBuffer[3] = minutes % 10; // Units place of minutes
    shiftOutBuffer[4] = seconds / 10;    // Tens place of seconds
    shiftOutBuffer[5] = seconds % 10;    // Units place of seconds

    update_display();
  }
}
```

Conclusion: The Digital clock made in this project is a simple Digital clock representing only hour, minute Second. it is a 24-hour clock. The construction of this clock is very simple.

References:

1. www.Youtube.com
2. https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html
3. https://www.coursehero.com/home/#/home/
4. https://friendtechbd.com