Defiarl'menl' of Sofl'ware Engineering

Submitted By :

Tanzeela Asghar(2021-BSE-032)
Neha Amjad(2021-BSE-024)
Laiba Sohail(2021-BSE-016)

Submitted To :

Sir Haroon Waseem

Course:

Embedded Systems

Date:

9th June 2024

"Presentation Assignment"

“Timer and Interrupts in AVR microcontrollers and I2C communication”

I, Tanzeela Asghar, solemnly declare that the work presented in my BSE project titled "Presentation" is solely my research work with no significant contribution from any other person or Generative AI Tools. Any small contributions or help taken have been duly acknowledged/cited, and the complete project has been written by me in accordance with the latest plagiarism policy declared by HEC and my respective university in line with the policy for use of Generative AI Tools.

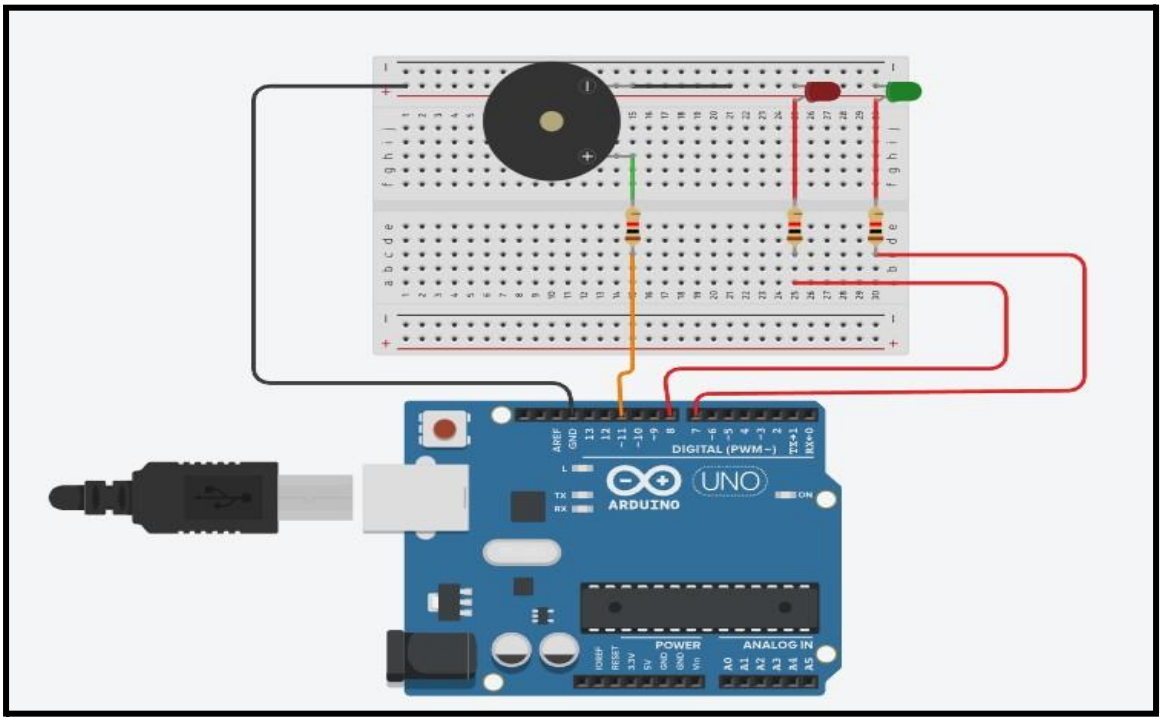
I understand the zero-tolerance policy of the HEC and FJWU towards plagiarism. Therefore, I, as the author of the above-titled assignment, declare that no portion of my project has been plagiarized, and any material used as a reference is properly referred/cited.

I undertake that if I am found guilty of any plagiarism in the above-titled project even after the award of my BSE degree, the University reserves the right to withdraw/revoke my degree. Furthermore, HEC and the University have the right to publish my name on the HEC/University website, where the names of students who submitted plagiarized projects are listed

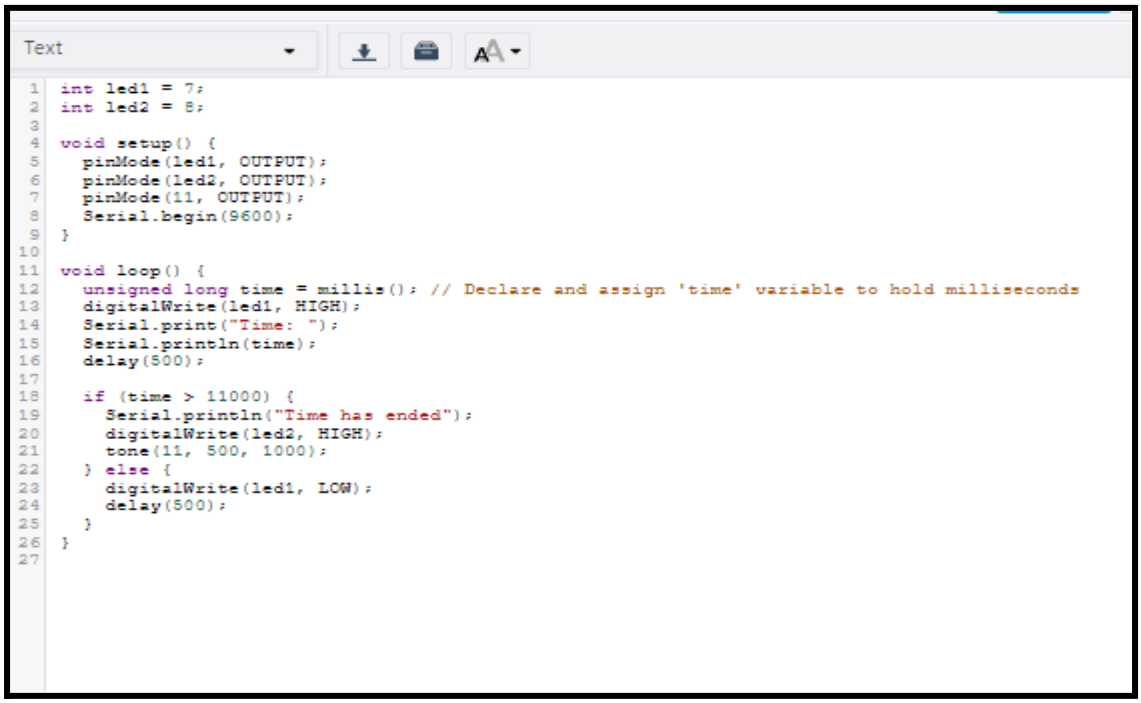
Timer:

Timers in AVR microcontrollers have input pins designed to trigger input capture events. When a signal change is detected at such a pin, the timer value is immediately read and stored in the Input Capture Register (ICRx). Concurrently, the Input Capture Flag (ICFx) in the Timer/Counter Interrupt Flag Register (TIFRn) is set. This feature is particularly useful for measuring the duration of external pulses.

Circuit diagram:



Code:



```
Text

int led1 = 7;
int led2 = 8;

void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(11, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  unsigned long time = millis(); // Declare and assign 'time' variable to hold milliseconds
  digitalWrite(led1, HIGH);
  Serial.print("Time: ");
  Serial.println(time);
  delay(500);

  if (time > 11000) {
    Serial.println("Time has ended");
    digitalWrite(led2, HIGH);
    tone(11, 500, 1000);
  } else {
    digitalWrite(led1, LOW);
    delay(500);
  }
}
```

Components:

- **Arduino Uno**: The main microcontroller board.
- **Breadboard**: For prototyping the circuit.
- **LEDs (Red and Green)**: Connected to digital pins 7 and 8 through resistors.
- **Piezo Buzzer**: Connected to digital pin 11 for sound output.
- **Resistors**: Current limiting resistors for LEDs.
- **Connecting Wires**: To establish connections between components and Arduino

Connections:

- **Red LED**: Anode connected to digital pin 8 via a resistor, cathode connected to ground.
- **Green LED**: Anode connected to digital pin 7 via a resistor, cathode connected to ground.
- **Buzzer**: Connected to digital pin 11 and ground.
- **Power and Ground Lines**: Connected from the Arduino to the breadboard to supply power to components

This cricut create using a simple timer with Arduino. The green LED blinks until 11 seconds have passed, after which the red LED lights up, and the buzzer sounds for 1 second. The current time is continuously printed to the serial monitor for debugging purposes.\

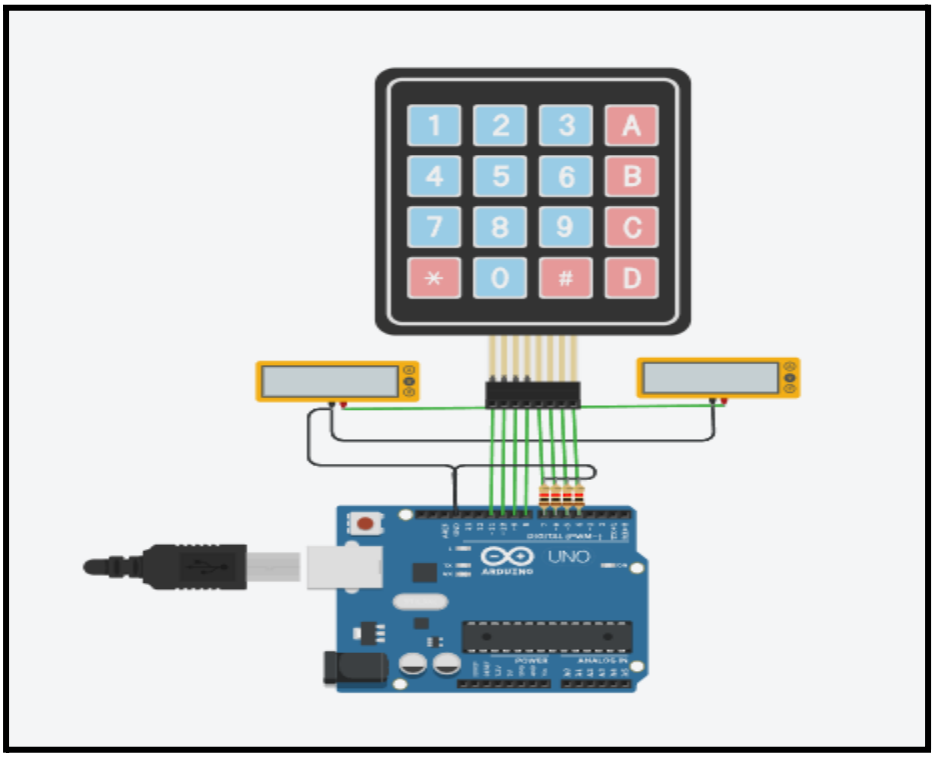
Link:

https://www.tinkercad.com/things/083r4zree3u-cool-lahdi/editel?returnTo=%2Fthings%2F083r4zree3u-cool-lahdi&sharecode=Xt7aOwVDOUbp7NmAR4TCjQL1fnXbct3iqIRgGBYQHPU

Interrupt:

AVR devices are equipped with external interrupts that can wake the device from sleep mode in response to a rising or falling edge signal or a change in the digital voltage level at an I/O pin. Once awakened, the device processes the application based on the interrupt source before returning to sleep mode.

Circuit diagram:



code:

```
#include <Keypad.h>

const byte ROWS = 4; // Four rows
const byte COLS = 4; // Four columns

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowPins[ROWS] = {2, 3, 4, 5};
byte colPins[COLS] = {6, 7, 8, 9};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  Serial.begin(9600);
}

void loop() {
  char key = keypad.getKey();

  if (key) { // Check if a key is pressed
    Serial.println(key);
    // Convert the character key to a voltage signal and print it
    float voltage = mapKeyToVoltage(key);
    Serial.print("Voltage: ");
    Serial.println(voltage);
  }
}

// Function to map keypad keys to a voltage signal (for demonstration)
float mapKeyToVoltage(char key) {
  // Here, we arbitrarily map each key to a specific voltage value
  // You can adjust these values to fit your needs or experiment with different mapping
  if (key >= '0' && key <= '9') {
    return (key - '0') * 0.1; // Map keys '0' to '9' to voltages from 0.0V to 0.9V
  } else if (key == '*') {
    return 1.0; // Map '*' key to 1.0V
  } else if (key == '#') {
    return 2.0; // Map '#' key to 2.0V
  } else {
    return 0.0; // Map other keys to 0.0V
  }
}
```

Rows (R1, R2, R3, R4) are connected to Arduino digital pins 2, 3, 4, 5.
Columns (C1, C2, C3, C4) are connected to Arduino digital pins 6, 7, 8, 9

The code sets up a 4x4 keypad and uses pin change interrupts to handle key presses. The keyPressed function is called whenever there is a change on one of the column pins.

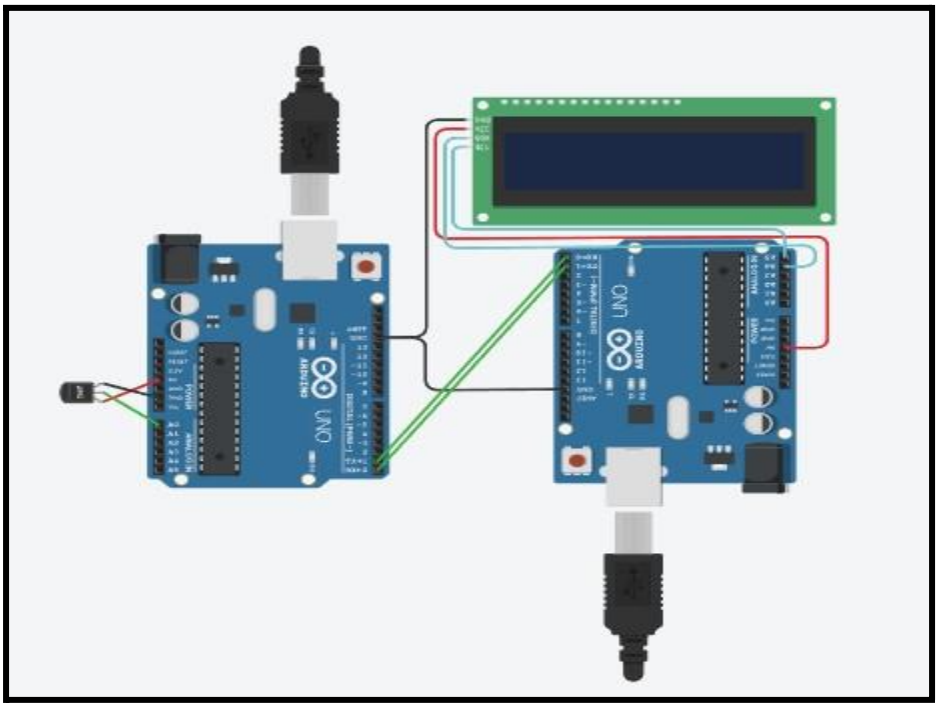
Link:

https://www.tinkercad.com/things/jDtO2sldphu-surprising-migelo-esboo/editel?sharecode=cgISpEfU46yfWznO2H2xJGrR6Yu4rcoiQm21AeGxBUQ

Spi and I2C communication:

I2C combines the best features of SPI and UART communication protocols. Like SPI, I2C allows you to connect multiple slave devices to a single master. Additionally, I2C supports multiple masters, enabling multiple microcontrollers to control one or more slaves. This feature is particularly useful when you need several microcontrollers to log data to a single memory card or display text on a single LCD.

Circuit diagram:



Components:

- **TMP36 Temperature Sensor**: Connected to an analog input pin (A0) on the Arduino.
- **Arduino Uno (Master)**: Reads the temperature and sends it over I2C
- **MCP23008-based LCD Module**: For I2C communication with the LCD.
- **Arduino Uno (Slave)**: Receives temperature data and displays it on the LCD.

Master Arduino:

Text 1 (Arduino Uno R3)

```
#include <Wire.h>

#define TMP36_PIN A0 // Analog pin connected to TMP36 sensor

void setup() {
  Wire.begin(); // Initialize I2C communication
  Serial.begin(9600); // Initialize serial communication
}

void loop() {
  // Read temperature from TMP36 sensor
  int sensorValue = analogRead(TMP36_PIN); // Read the temperature sensor value
  float voltage = sensorValue * (5.0 / 1023.0); // Convert the analog reading to voltage (0 to 5V)
  float temperatureC = (voltage - 0.5) * 100; // Convert voltage to temperature (in degrees Celsius) for TMP36 sensor

  // Send temperature data over I2C to slave Arduino
  Wire.beginTransmission(0x27); // Slave address (0x27 for example, adjust as needed)
  Wire.write((byte*)&temperatureC, sizeof(temperatureC)); // Send temperature data
  Wire.endTransmission(); // Stop transmitting

  Serial.println(temperatureC); // Print temperature to serial monitor

  delay(1000); // Wait for 1 second before the next reading
}
```

Slave Aduino:

Text

```
#include <Wire.h>
#include <LiquidCrystal.h>

#define LCD_ADDRESS 0x20 // Address of the MCP23008-based LCD module

#define LCD_COLUMNS 16 // Number of columns in the LCD
#define LCD_ROWS 2     // Number of rows in the LCD

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // Initialize LCD

void setup() {
  Wire.begin(9); // Initialize I2C communication as Slave with address 9
  Wire.onReceive(receiveData); // Register callback for receiving data
  Serial.begin(9600); // Initialize serial communication
  lcd.begin(LCD_COLUMNS, LCD_ROWS); // Initialize LCD
  lcd.print("Temperature: "); // Print static message
}

void loop() {
  // Do other stuff if needed, but keep it non-blocking
}

void receiveData(int byteCount) {
  float temperatureC;
  while (Wire.available()) {
    Wire.readBytes((uint8_t*)&temperatureC, sizeof(temperatureC)); // Receive temperature data
  }

  Serial.println(temperatureC); // Print received temperature to serial monitor

  lcd.setCursor(0, 1); // Set cursor to the beginning of the second row
  lcd.print("Temp: ");
  lcd.print(temperatureC); // Print the temperature
  lcd.print(" C   ");      // Print units and clear any leftover characters
}
```

- The master Arduino reads temperature data from a TMP36 sensor and sends it via I2C to a slave Arduino.
- The slave Arduino receives the temperature data and displays it on an LCD screen.
- The master Arduino also prints the temperature data to the serial monitor for debugging

Link:

https://www.tinkercad.com/things/441pkEaWHto-ingenious-habbi/editel?sharecode=CzTJMM8OWlVnZWrvS1xqrFUQ506tEt2kqEqBR1aL8jM