Assignment Module 2 - Part B

IOT internship

Team Members: -

TL: Mahmoud Essam Fathy: 20221460231
Ziad Ashraf Hafez Gaber: 20221374025
Ziad Ashraf Ibrahim Taher: 20221369225

Assignment – Module 2 – Part B – Theoretical

< Mention the usage of interrupts and timers in ESP32 project and why It is important? >

<Answer>

Interrupts	Timer
Work in the context of microcontrollers like the ESP32. The attachInterrupt() function in the Arduino IDE is used to set up an interrupt, and it takes the GPIO pin, the name of the function to be executed, and the interrupt mode as arguments. When an interrupt occurs, the processor stops executing the main program and jumps to the interrupt handler function, or ISR, which handles the event. This allows the microcontroller to respond promptly to external events without having to continuously poll for them in the main program loop	Implement a motion detection system with a PIR sensor and an LED, we can use the milis() method as a timer to turn the LED on for a specific amount of time after motion is detected. This approach is preferred over using the delay() method, which would pause the program execution for a specified time instead of allowing it to continue running while the timer counts down. By using millis(), the program can continue to execute and perform other tasks while the LED is on, which can be useful in more complex applications.

Timers and interrupts

Are important features of the ESP32 microcontroller?
Timers can be used to generate periodic events or measure time intervals with high accuracy, while interrupts allow for quick responses to external events without having to continuously poll for them in the main program loop

Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

Assignment – Module 2 – Part B – Theoretical

The ESP32 can interface with an LCD display using various communication protocols

<Answer>

The actual connections and communication methods depend on the specific LCD module you are using. Most LCD modules come with their own datasheets that provide details about the required connections and communication protocols. When working with the ESP32, you can use libraries like the "Adafruit GFX" and "Adafruit ILI9341" to interface with popular LCDs like ILI9341 that support SPI communication. For I2C communication, libraries like "Adafruit SSD1306" can be used for OLED displays.

How to connect LCD "No I2C" to esp32?

1. **Identify the LCD Module:** Determine the specific model and type of your LCD display and obtain its datasheet.
2. **Make the Physical Connections:** Connect the necessary pins between the ESP32 and the LCD module based on the chosen communication protocol (e.g., SPI, I2C, or Parallel).
3. **Install Required Libraries:** Depending on the LCD type and communication protocol, install the appropriate libraries in your Arduino or ESP-IDF environment.
4. **Initialize the LCD:** Initialize the LCD and configure its settings, such as resolution, orientation, and color mode.
5. **Write Data to the LCD:** Use the library functions to display text, graphics, or other content on the LCD screen.
6. **Test and Debug:** Upload the code to the ESP32 and test the display. If necessary, troubleshoot and debug any issues that may arise.

Parallel Interface	Serial Peripheral	Inter-Integrated Circuit (I2C)
The parallel interface involves connecting multiple data pins (usually 8 or 16) and control pins (like RS, R/W, and E) between the ESP32 and the LCD. This method allows for faster data transfer but requires more GPIO pins, which might be a limitation on some ESP32 boards.	SPI is a synchronous serial communication protocol that uses four lines for communication: SCLK (Serial Clock), MOSI (Master Out Slave In), MISO (Master In Slave Out), and SS/CS (Slave Select/Chip Select). This protocol is widely used and is supported by most LCD modules with an SPI interface	I2C is another serial communication protocol that uses two lines: SDA (Serial Data) and SCL (Serial Clock). I2C is straightforward to use and requires only two GPIO pins, making it suitable for scenarios with limited pins available.

Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

Assignment – Module 2 – Part B – Theoretical

< How can an IR sensor be used to detect objects, and what is the >
< Principle behind its operation? >

<Answer>

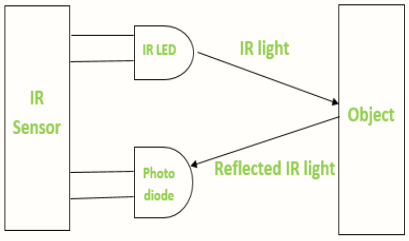
IR sensor in general is electronic device, that emits the light in order to sense some object of the surroundings

Its process based on 2 parts, IR Transmitter and IR Reciever

IR Transmitter : Infrared Transmitter is a light emitting diode (LED) which emits infrared radiations called as IR LED's. Even though an IR LED looks like a normal LED, the radiation emitted by it is invisible to the human.

IR Reciever : detect the radiation from an IR transmitter

Active infrared sensors both emit and detect infrared radiation.



When an object comes close to the sensor, the infrared light from the LED reflects off of the object and is detected by the receiver.

Based on the signals come from Receiver, it performs action as if the object was in front Of the led, distance will be so near, and by that will perform specific action till it backs to Normal shape

Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

Assignment – Module 2 – Part B – Theoretical

PWM (Pulse Width Modulation) signals can be generated using the ESP32 microcontroller by utilizing its built-in hardware PWM capabilities.

<Answer>

The ESP32 has several PWM channels that can be used to produce PWM signals on specific GPIO pins. Here's how you can generate PWM signals using the ESP32:

1.Choose a PWM Channel: First, decide which PWM channel you want to use. The ESP32 has multiple PWM channels, and each channel is associated with specific GPIO pins. Refer to the ESP32 documentation or your development board's pinout diagram to identify the available PWM channels and their corresponding GPIO pins.
2. PWM Frequency and Duty Cycle: After selecting the PWM channel, you can configure the PWM frequency and duty cycle. The frequency determines how fast the PWM signal oscillates, while the duty cycle represents the ratio of the high signal time to the total period (high + low) of the PWM signal.
3.Initialize PWM: Use the appropriate functions from the ESP32's API to initialize the selected PWM channel with the desired frequency and duty cycle settings.
4.Start/Stop PWM: Once the PWM channel is initialized, you can start and stop the PWM output as needed using the appropriate API functions.

Now, let's consider the main factors to consider when configuring PWM:

```
const int LED = 18;  /*Equals to GPIO pin 18*/
const int freq = 5000; /*PWM signal frequency*/
const int LED_Channel = 0;
const int resolution = 8; /*PWM resolution*/
void setup(){
  ledcSetup(LED_Channel, freq, resolution);  /*PWM signal defined*/
  ledcAttachPin(LED, LED_Channel);
}
void loop(){
  for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){   /*LED brightness increases*/
    ledcWrite(LED_Channel, dutyCycle);
    delay(15);
  }
  for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){   /*LED brightness decreases*/
    ledcWrite(LED_Channel, dutyCycle);
    delay(15);
  }
}
```

5. Frequency: The PWM frequency determines how fast the PWM signal oscillates. Higher frequencies can provide smoother control for some applications but might result in audible noise if used for driving motors or other mechanical devices. Lower frequencies, on the other hand, might introduce flickering in some applications. Choose a frequency appropriate for your specific application and the characteristics of the peripheral you are controlling.
6. Duty Cycle: The duty cycle represents the ratio of the high signal time to the total period of the PWM signal. It is expressed as a percentage. A 50% duty cycle means the signal is on for half the time and off for the other half. The duty cycle determines the "average" voltage or current applied to the load. It is used to control the brightness of LEDs or the speed of motors, for example.
7. Resolution: The resolution of the PWM signal determines the number of discrete steps between fully off and fully on. The ESP32 supports different bit resolutions, typically 8 or 10 bits, depending on the specific PWM channel. Higher resolution provides finer control but may require more CPU resources.
8. GPIO Compatibility: Check the compatibility of the chosen GPIO pin with PWM. Not all GPIO pins on the ESP32 support PWM output.

Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

5. Current Limitations: Ensure that the GPIO pin and the connected circuit can handle the current required by the device being controlled. Exceeding the current limitations may damage the GPIO pin or the connected components.
6. Interference: Consider potential interference from other PWM signals or external noise sources that might affect the performance of the PWM output or the connected devices.

By taking these factors into account, you can configure PWM signals effectively for various applications using the ESP32 microcontroller.

Assignment – Module 2 – Part B – Theoretical

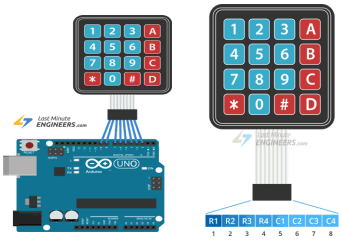
< Discuss the principle behind matrix keypads and how they can be interfaced with the ESP32> < for user input >

<Answer>

1. Keypad: Matrix keypads consist of a grid of buttons with corresponding wires that may be read and interpreted by a microcontroller.
2. To detect which button has been pressed, the keypad controller scans the rows and columns in a sequential manner. It applies a voltage to one row at a time, and reads the voltage on each column. If a button is pressed, it will create a connection between the row and column, and the voltage at the corresponding column will change. By scanning through all the rows and columns, the controller can determine which button has been pressed

What about how to connect them and interface in ESP32?
-> Only need is about 8 pins, from 1 to 4 "Rows" and 5 to 8 "columns"
-> Configure the GPIO pins as input pins in the ESP32 code.
-> Scan the matrix keypad by iterating through each row and setting the corresponding row pin to high, then reading the state of each
-> Column pin to determine which button has been pressed.
-> Then after connection them all printing your own results on Serial monitor or try print it
-> Using LCD if connected.

```
byte rowPins[ROWS] = {R1, R2, R3, R4}; /* connect to the row pinouts of the keypad */
byte colPins[COLS] = {C1, C2, C3, C4}; /* connect to the column pinouts of the keypad */
```



Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

Module 3 Assignment Part A – ESP32 Problem 1 (Door)

Description of connections

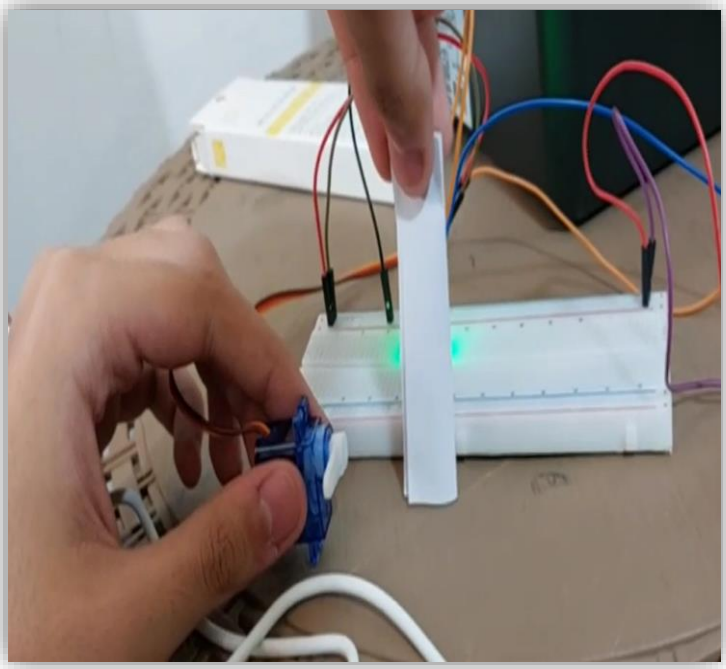
A- The code uses an ESP32 microcontroller, an IR sensor, and a servo motor to control the opening and closing of a door based on the presence of a person.

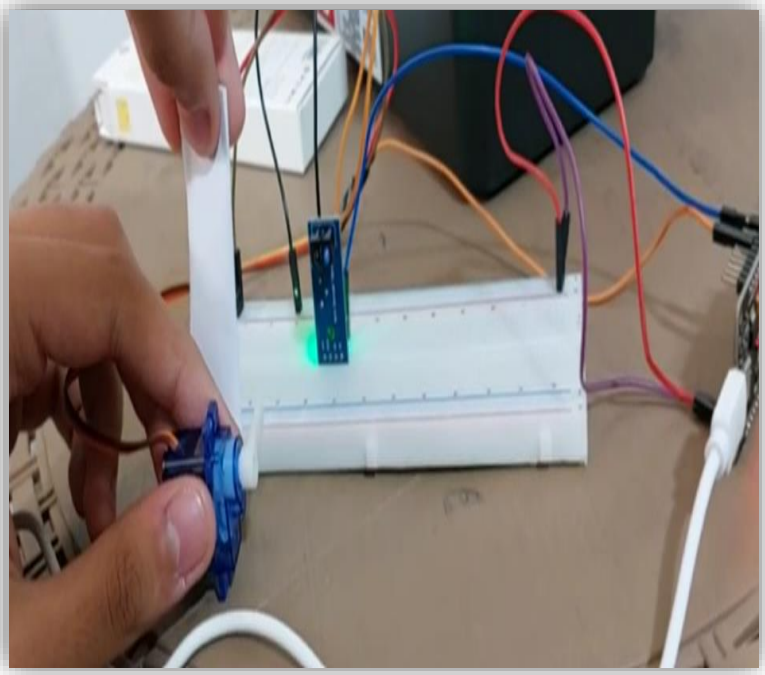
- The IR sensor should be connected to an analog input pin on the ESP32, which in this code is defined as pin 32.
- The servo motor should be connected to a PWM output pin on the ESP32, which in this code is defined as pin 26.

The void setup() : attaches the servo motor to its corresponding pin, and sets the initial angle of the servo motor to 0 degrees to indicate that the door is closed.

The void loop() : function reads the analog value from the IR sensor using analogRead(), and sets the angle of the servo motor to open or close the door accordingly using doorServo.write().

Used materials : IR : for detecting distance, Servo : as door
Everything Provided in the folder "Videos – Circuits – Schematic – Real-life – and TinkerCAD action"





Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

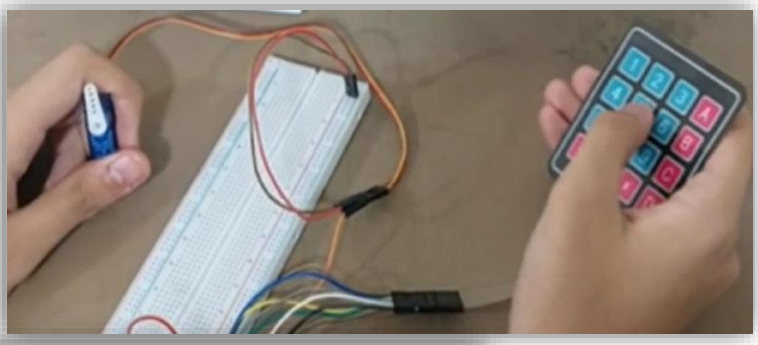
Module 3 Assignment Part B – ESP32 Keypad Angles

Description of connections

The code uses an ESP32 microcontroller, a keypad, and a servo motor to control the rotation angle of the servo motor based on input from the keypad.

in the void setup() function : attaches the servo motor to the corresponding pin using myServo.attach(), and sets the initial position of the servo motor to 0 degrees using myServo.write().

In the void loop() function: the program reads input from the keypad using keypad.getKey(), and sets the angle of the servo motor based on the key pressed by the user using myServo.write(),
If the user presses the key '1', the program calls myServo.write(30) to rotate the servo motor to 30 degrees.
If the user presses the key '*', the program calls myServo.write(0) to rotate the servo motor to stop position.
If the user presses the key 'D', the program calls myServo.write("User angle") to rotate the servo motor to the angle, user will put in serial.
Used materials : Keypad : For controlling angles, Servo : for targeting the needed angles
Everything Provided in the folder "Videos – Circuits – Schematic – Real-life – and TinkerCAD action"



Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

Module 3 Assignment
Part B – ESP32
IR Distance LCD

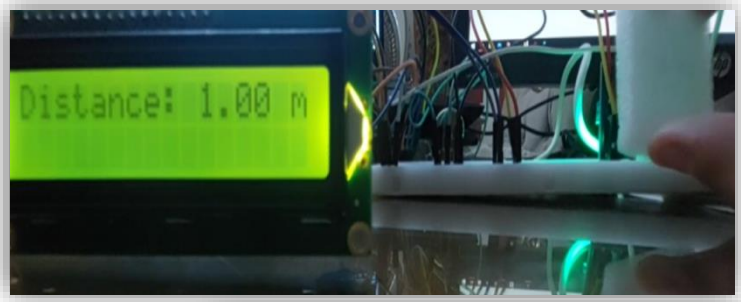
Description of connections

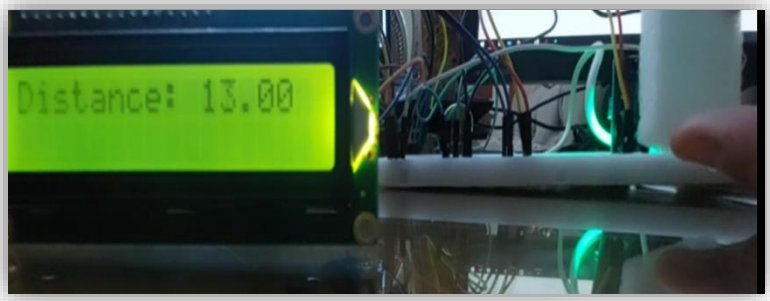
This code uses an IR sensor and an LCD screen to measure the distance of an object from the sensor and display the value on the screen. Here's a brief explanation of the code:

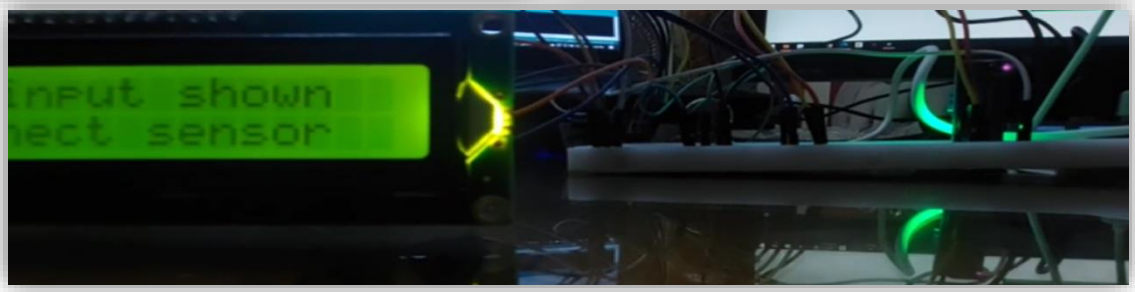
- The code defines a constant variable IR_SENSOR which is connected to the analog input pin of the IR sensor.
- The code also initializes the LiquidCrystal library to interface with a 16x2 LCD screen using the pins rs, e, d4, d5, d6, and d7 connected to the LCD screen.

In the setup() function : the LCD screen is initialized with a size of 16x2.
In the loop() function : the analog value from the IR sensor is read using the analogRead() function and stored in the variable irValue.
If the irValue is less than 10, it implies that the IR sensor is not connected or has a poor connection. In this case, an error message is displayed on the LCD screen indicating that the sensor is not connected.
If the irValue is greater than or equal to 10, it means that the sensor is connected and working. In this case, the analog value is mapped to a distance value between 1-25 mm using the map() function and stored in the variable distance.
The distance value is then displayed on the LCD screen using the lcd.print() function.
And if something wrong happened, will print it in LCD as handling for error

Used Material : LCD to print results, IR sensor to calculate distance







Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

Module 3 Assignment Part A – ESP32 Quick Bonus (GAS)

Description of connections

This project is designed to detect gas concentration using an analog gas sensor and trigger an alarm if the concentration exceeds a certain threshold. The project utilizes an ESP32 DevKit V1 board, a gas sensor, an LED, a buzzer, and an LCD display.

The gas sensor is connected to analog pin 34 on the ESP32 DevKit V1 board. The LED is connected to pin 13 on the board, and the buzzer is connected to pin 12. The LCD display is connected to the following pins on the board: RS pin to pin 22, E pin to pin 23, D4 pin to pin 5, D5 pin to pin 18, D6 pin to pin 19, and D7 pin to pin 21. Make sure to also connect the power and ground pins for each component, and use appropriate resistors if necessary.

E. Used components and what they are used for in this circuit:

ESP32 DevKit V1 board: This is a microcontroller board that is used to control the other components in the circuit.

Gas sensor: This is an analog sensor that is used to detect gas concentration.

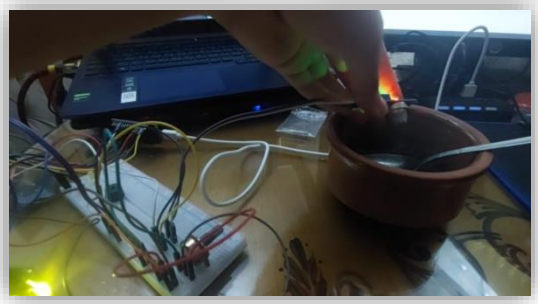
LED: This component is used to indicate the status of the gas concentration (on or off).

Buzzer: This component is used to sound an alarm if the gas concentration exceeds a certain threshold.

LCD display: This component is used to display messages related to the gas concentration.

Overall, this project is a great example of how to use different components in combination to achieve a specific goal.

Everything Provided in the folder "Videos – Circuits – Schematic – Real-life – and TinkerCAD action"



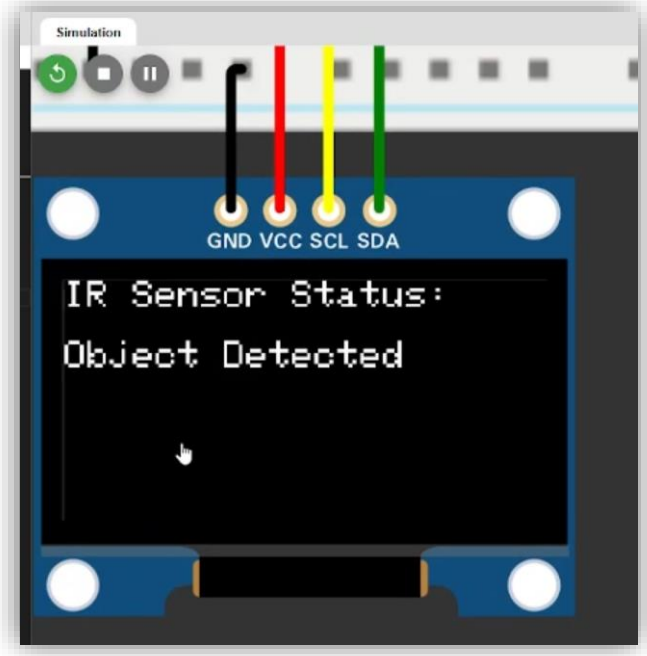
Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

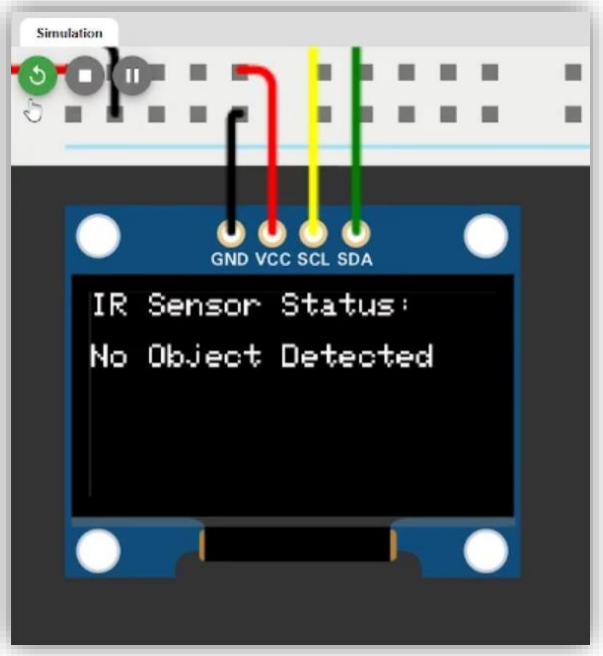
Module 3 Assignment Part B – ESP32 OLED

Description of connections

- This project is designed to detect the presence of an object using an infrared (IR) sensor and display the status on an OLED display. The project utilizes an ESP32 DevKit V1 board, an IR sensor, and an OLED display.
- The IR sensor is connected to pin 33 on the ESP32 DevKit V1 board. No additional components are required as the sensor can be directly interfaced with the board. The OLED display is connected to the board using I2C communication, with SDA connected to pin 21 and SCL connected to pin 22.
- E. Used components and what they are used for in this circuit:
- ESP32 DevKit V1 board: This is a microcontroller board that is used to control the other components in the circuit.
- IR sensor: This is a sensor that is used to detect the presence of an object by emitting and detecting infrared radiation.
- OLED display: This component is used to display the status of the IR sensor. It is connected to the board using I2C communication.

Everything Provided in the folder "Videos – Circuits – Schematic"





Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

Module 3 Assignment Part A – ESP32 Quick Bonus (RAIN)

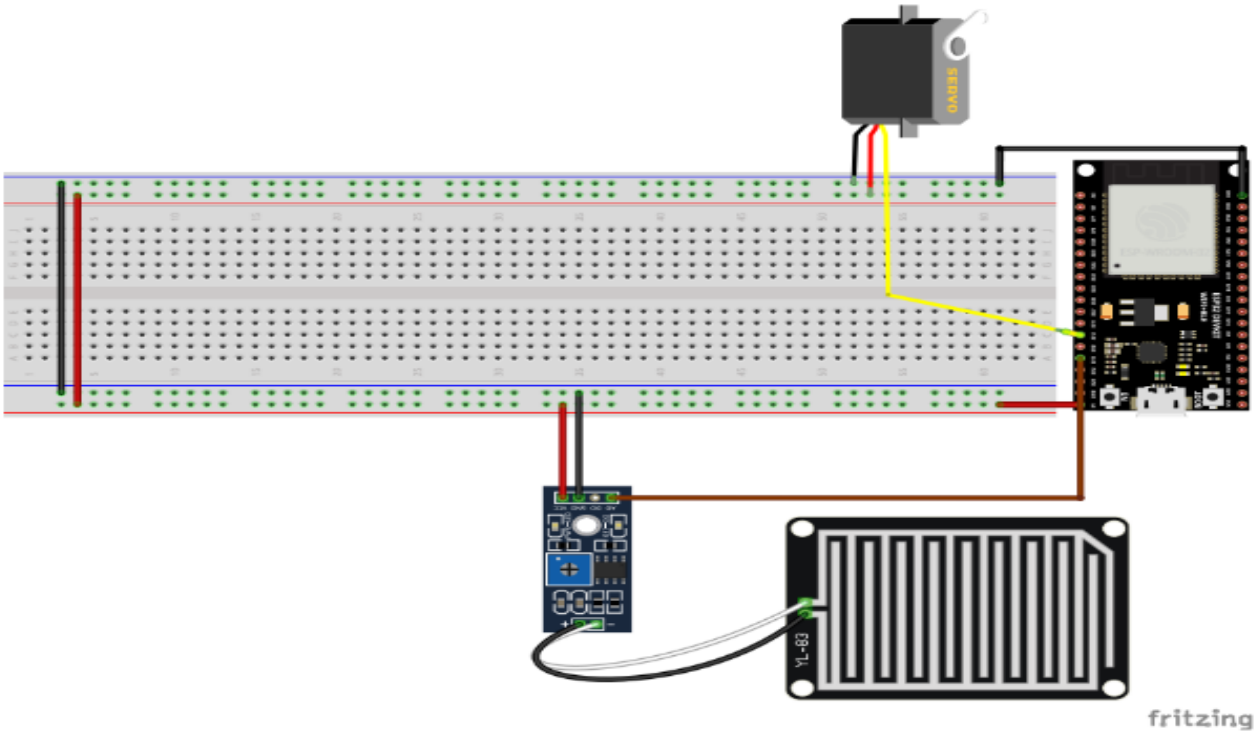
Description of connections

This project is designed to detect rain using a rain sensor and control a servo motor to close a window if it starts to rain. The project utilizes an ESP32 board, a rain sensor module, and a servo motor.

The rain sensor module is connected to pin 34 on the ESP32. The servo motor is connected to pin 15 on the board. No additional components are required as the rain sensor module and servo motor can be directly interfaced with the board.

ESP32: This is a microcontroller board that is used to control the other components in the circuit.
Rain sensor module: This is a sensor that is used to detect rain by measuring the resistance between its two electrodes.
Servo motor: This component is used to control the opening and closing of a window. It is connected to the board and controlled using the Servo library.

Sadly we made design, because Rain sensor isn't available in all simulators



Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

Module 3 Assignment Part B – ESP32 Temp Hum

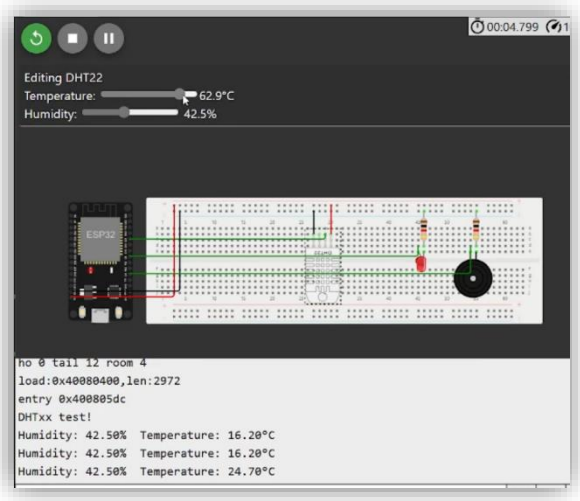
Description of connections

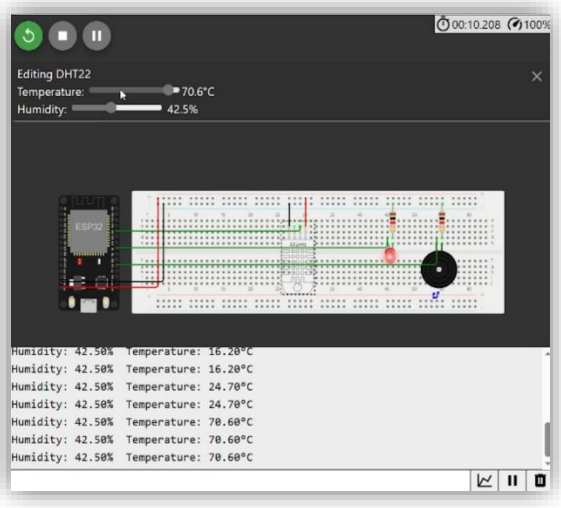
This project is designed to detect rain using a rain sensor and control a servo motor to close a window if it starts to rain. The project utilizes an ESP board, a rain sensor module, and a servo motor.

The rain sensor module is connected to pin 13 on the ESP board. The servo motor is connected to pin 3 on the board. No additional components are required as the rain sensor module and servo motor can be directly interfaced with the board.

- ESP board: This is a microcontroller board that is used to control the other components in the circuit.
- Rain sensor module: This is a sensor that is used to detect rain by measuring the resistance between its two electrodes.
- Servo motor: This component is used to control the opening and closing of a window. It is connected to the board and controlled using the Servo library.

Everything Provided in the folder "Videos – Circuits – Schematic"





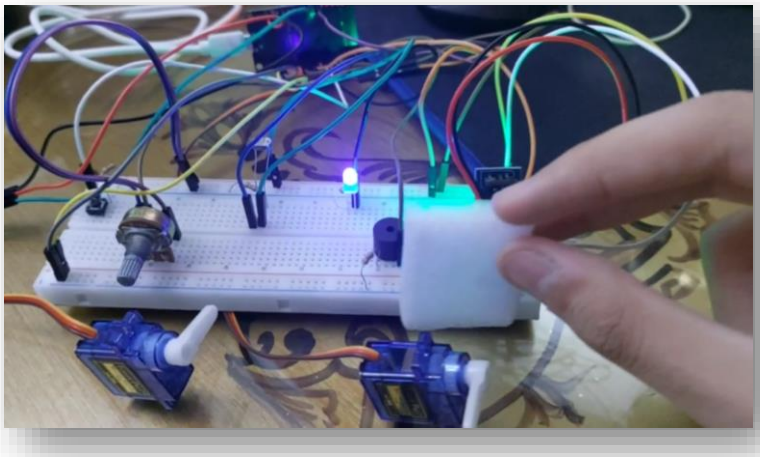
Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem

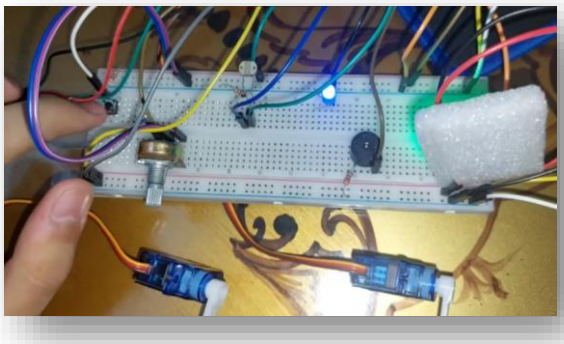
Module 3 Assignment Part A – ESP32 Bonus (HOME)

Description of connections

A. This project appears to be a basic home automation system that uses an ESP32 microcontroller to control a servo motor for a door and a window, as well as an LED, buzzer, and two sensors (an infrared sensor and an LDR) to detect and respond to certain events. The system includes a reset button that can be used to return all components to their default state.

- IR sensor (pin 25): This is an input pin that is used to detect the presence of an object in front of the infrared sensor.
- LED (pin 26): This is an output pin that is used to control the LED.
- Servo motor for door (pin 19): This is a digital output pin that is used to control the position of the door servo motor.
- Servo motor for window (pin 18): This is a digital output pin that is used to control the position of the window servo motor.
- Buzzer (pin 27): This is an output pin that is used to control the buzzer.
- Potentiometer (pin 34): This is an input pin that is used to read the analog value of the potentiometer.
- LDR (pin 35): This is an input pin that is used to read the analog value of the LDR.
- Reset button (pin 32): This is an input pin that is used to detect when the reset button is pressed.
- ESP32 microcontroller: This is the main microcontroller that controls the operation of the system.
- IR sensor: This is an infrared sensor that is used to detect the presence of an object in front of it.
- LED: This is a simple LED that is used to indicate the status of the system.
- Servo motor for door: This is a servo motor that is used to control the position of the door.
- Servo motor for window: This is a servo motor that is used to control the position of the window.
- Buzzer: This is a simple buzzer that is used to indicate the status of the system.
- Potentiometer: This is a simple potentiometer that is used to adjust the position of the window servo motor.
- LDR: This is a Light Dependent Resistor that is used to detect the ambient light level.
- Reset button: This is a simple push button that is used to reset the system to its default state.





Mahmoud Essam, Ziad Ashraf, Ziad Ashraf
Supervised by: Eng. Muhammed Hatem