MICROPROCESSORS AND MICROCONTROLLERS LABORATORY-21EC211

Title of the Project: SMART LOCKER

MINI PROJECT – REPORT

Abstract (10)	
H/W & S/W description (30)	
Block diagram & Working (30)	
Working Model (10)	
Applications (10)	
Results (10)	
Total (100)	

Name of the students:

Imayavaramban– 21ECE039

Harish– 21ECE038

Janarthanan– 21ECE041

Gurunathan-21ECE037

Course In charge: Dr. S. Gandhimathi @ Usha

Date of Submission: 9 May, 2023

Abstract:

This project involves using an ultrasonic sensor to measure the distance between the sensor and an object. When the distance is less than a certain threshold (10cm in this case), a message is sent to a specified phone number using the Twilio API to notify the recipient that a locker or other object has been opened.

Apparatus used:

- Raspberry pi pico w
- Ultrasonic sensor
- USB cable

Description:

Hard ware description:

- Raspberry Pi Pico W microcontroller
- HC-SR04 ultrasonic sensor
- USB cable for power and programming

Raspberry Pi Pico microcontroller:

Raspberry Pi Pico W adds on-board single-band 2.4GHz wireless interfaces (802.11n) using the Infineon CYW43439 while retaining the Pico form factor. The on-board 2.4GHz wireless interface has the following features:

- Wireless (802.11n), single-band (2.4 GHz)
- WPA3
- Soft access point supporting up to four clients

The antenna is an onboard antenna licensed from ABRACON (formerly ProAnt). The wireless interface is connected via SPI to the RP2040 microcontroller.

Due to pin limitations, some of the wireless interface pins are shared. The CLK is shared with VSYS monitor, so only when there isn't an SPI transaction

in progress can VSYS be read via the ADC. The Infineon CYW43439 DIN/DOUT and IRQ all share one pin on the RP2040. Only when an SPI transaction isn't in progress is it suitable to check for IRQs. The interface typically runs at 33MHz.

For best wireless performance, the antenna should be in free space. For instance, putting metal under or close by the antenna can reduce its performance both in terms of gain and bandwidth. Adding grounded metal to the sides of the antenna can improve the antenna's bandwidth.

HC-SR04 ultrasonic sensor:

The HC-SR04 is a popular ultrasonic distance sensor that can measure distances up to 4 meters. It is commonly used in robotics and automation projects for obstacle detection and avoidance.

The sensor works by emitting a high-frequency sound wave from its transmitter, and then measuring the time it takes for the sound wave to bounce off an object and return to the receiver. By knowing the speed of sound, the time it takes for the sound wave to return can be used to calculate the distance between the sensor and the object.

The HC-SR04 sensor consists of four pins: Vcc, Trig, Echo, and Gnd. The Vcc pin is used to power the sensor, and should be connected to a 5V power supply. The Gnd pin is the ground connection. The Trig pin is used to trigger the sensor to start a measurement, and should be connected to a digital output pin on a microcontroller board. The Echo pin is used to receive the ultrasonic signal and measure the time it takes for the signal to return, and should be connected to a digital input pin on a microcontroller board.

To use the HC-SR04 sensor, a microcontroller board (such as Arduino, Raspberry Pi, or ESP8266) is typically used to send a trigger signal to the sensor, wait for the ultrasonic signal to return, and then calculate the distance using the measured time. The sensor can be programmed using a variety of programming languages and libraries.

 Software description:

MicroPython programming language

Network module for connecting to Wi-Fi

'urequests' module for sending HTTP requests to the Twilio API

Time module for timing and delays

Micropython:

MicroPython is a version of the Python programming language that is optimized to run on microcontrollers and small embedded systems. It provides a way to program microcontrollers using the Python language, which is known for its simplicity, readability, and ease of use.

MicroPython includes a subset of the Python language, with a focus on the core language features needed for microcontroller programming. It also includes a set of built-in modules that provide access to common microcontroller peripherals such as GPIO, I2C, SPI, and UART. These modules provide an easy way to interface with external hardware devices.

MicroPython can be installed on a wide range of microcontroller boards, including the popular ESP32 and ESP8266 Wi-Fi boards, the BBC micro:bit, and the Raspberry Pi Pico. Once installed, the board can be connected to a computer and programmed using a text editor and a serial connection.

MicroPython has become increasingly popular in the maker and hobbyist communities, as it provides an easy way to create interactive projects and prototypes using low-cost microcontrollers. It is also used in industrial applications for prototyping and proof-of-concept testing.

UREQUESTS:

urequests is a MicroPython library for making HTTP requests. It is a simplified version of the requests library for Python, and is designed to work with MicroPython's limited resources and capabilities.

The urequests library allows a MicroPython device to communicate with web servers and web services by sending HTTP requests and receiving responses. It supports various HTTP methods such as GET, POST, PUT, and DELETE, and can also handle HTTP authentication, headers, and data encoding.

urequests is built on top of MicroPython's built-in socket library, which provides a low-level interface for sending and receiving data over the network. It uses this library to establish a connection to the web server and send the HTTP request, and then receives the response and returns it to the calling program.

Using urequests, a MicroPython device can interact with web services such as REST APIs and webhooks, and can also be used to retrieve data from websites and online databases. It can be especially useful in IoT and sensor applications, where data from sensors can be sent to web services for processing and analysis.

NETWORK AND TIME MODULE:

The `network` module and `time` module in MicroPython for Raspberry Pi Pico are used for handling network connectivity and time-related operations respectively.

The `network` module provides functions to create and manage network connections, including Wi-Fi and Ethernet connections. With this module, you can connect your Raspberry Pi Pico to the internet or other network devices and send or receive data.

Here are some of the functions provided by the `network` module:

- `WLAN`: This function is used to create a WLAN (Wireless Local Area Network) object, which represents a wireless netwo*rk connection. It can be used to connect to a Wi-Fi

network, set the network parameters, and get the status of the connection.

- `LAN`: This function is used to create a LAN (Local Area Network) object, which represents an Ethernet connection. It can be used to set the network parameters and get the status of the connection.

The `time` module provides functions to manipulate time-related values such as the current time, time zones, and time intervals. With this module, you can get the current time, sleep for a certain duration, or calculate time differences between two timestamps.

Here are some of the functions provided by the `time` module:

- `sleep`: This function is used to pause the execution of the program for a specified number of seconds.

- `ticks_ms`: This function returns the number of milliseconds since the program started.

- `ticks_diff`: This function calculates the difference between two timestamps, in milliseconds.

Together, the `network` and `time` modules allow for easy network connectivity and time handling in MicroPython on the Raspberry Pi Pico, making it a versatile platform for IoT and other networked applications.

TWILIO:

Twilio is a cloud communications platform that provides a range of APIs and services to help businesses communicate with their customers via SMS, voice, video, and chat. The platform is designed to make it easy for developers to integrate communication functionality into their applications.

Twilio provides a number of APIs, including:

- Programmable SMS: Allows developers to send and receive SMS messages from their applications.

- Programmable Voice: Allows developers to add voice calling functionality to their applications.
- Programmable Video: Allows developers to add video calling functionality to their applications.
- Programmable Chat: Allows developers to add chat functionality to their applications.

Twilio also provides a number of services, including:

- *Phone numbers: Allows businesses to purchase and manage phone numbers for their applications.
- Authentication: Allows businesses to verify the identity of their users using two-factor authentication and other methods.
- Messaging services: Allows businesses to manage their SMS messaging campaigns and track their effectiveness.
- TaskRouter: Allows businesses to manage their call centers and distribute calls to agents based on their skills and availability.

To use Twilio, developers must sign up for a Twilio account, purchase a phone number, and obtain an authentication token. They can then use Twilio's APIs and services to send and receive messages, make and receive calls, and add other communication functionality to their applications.

Code:

```
import time
import network
import urequests


def send_sms(ssid, password, recipient, sender,
             message, auth_token, account_sid):
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wl*an.connect(ssid, password)
    while max_wait > 0:
```

```
    if wlan.status() < 0 or wlan.status() >= 3:
      break
    max_wait -= 1
    print('waiting for connection...')
    time.sleep(1)
  # Handle connection error
  if wlan.status() != 3:
     raise RuntimeError('network connection failed')
  else:
    print('connected')
    status = wlan.ifconfig()
    #print('ip = ' + status[0])

  headers = {'Content-Type': 'application/x-www-form-urlencoded'}
  data = "To=" + recipient + "&From=" + sender + "&Body=" + message
  print("Attempting to send SMS")
  r = urequests.post("https://api.twilio.com/2010-04-01/Accounts/" +
                     account_sid + "/Messages.json",
                     data=data,
                     auth=(account_sid,auth_token),
                     headers=headers)
  if r.status_code >= 300 or r.status_code < 200:
      print("There was an error with your request to send a message. \n" +
            "Response Status: " + str(r.status_code))
  else:
      print("Success")
      print(r.status_code)
  r.close()
import machine
import time

trigger = machine.Pin(2, machine.Pin.OUT)
echo = machine.Pin(3, machine.Pin.IN)

while True:
  trigger.low()
  time.sleep_us(2)
  trigger.high()
  time.sleep_us(10)
  trigger.low()

  while echo.value() == 0:
```

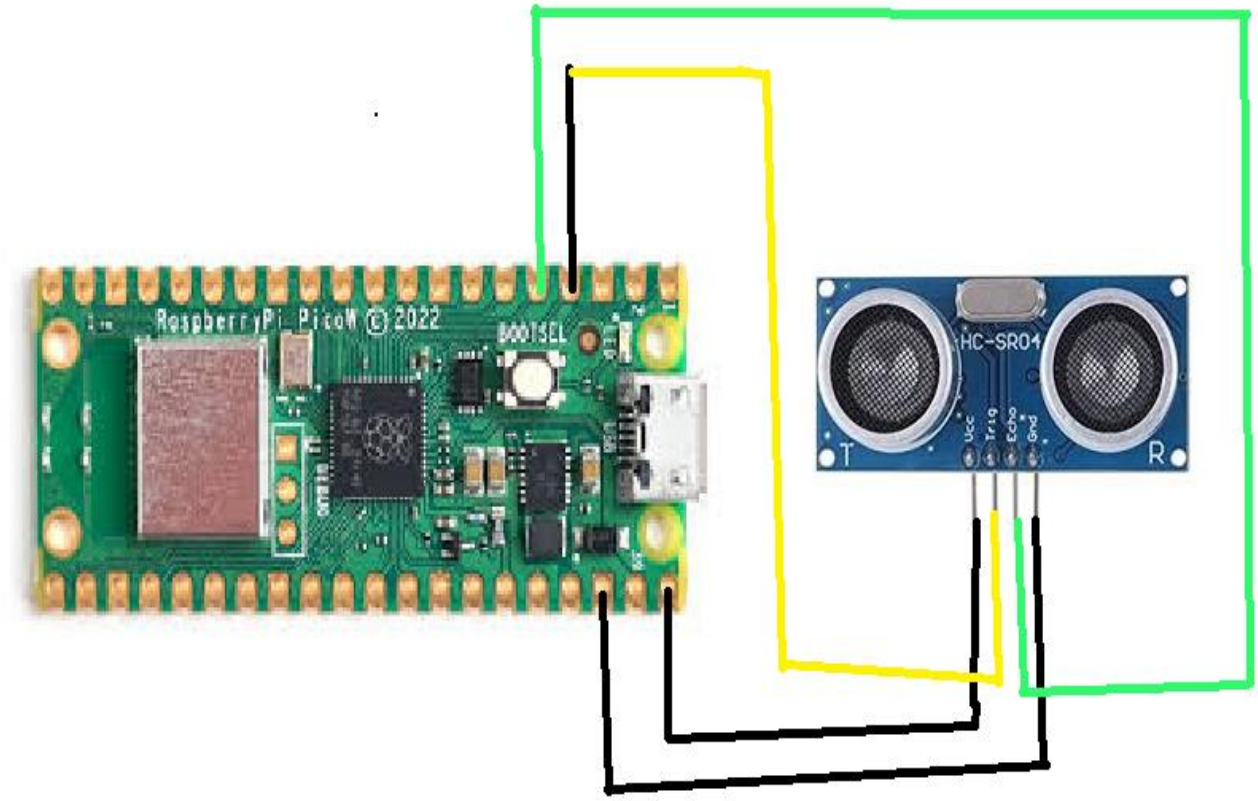
```
        pass
    pulse_start = time.ticks_us()

    while echo.value() == 1:
        pass
    pulse_end = time.ticks_us()
    +
    P*ulse_duration = pulse_end - pulse_start
    distance = pulse_duration / 58

    print("Distance:", distance, "cm")
    time.sleep(1)
    if(distance>10):
        send_sms('OPPO Reno6 5G','jana
kutty','+918610058375','+12707478710','locker was
opened','b9099551105ef3b3e26cef262b0274b6','ACbaeb9ad7e41a9ef9b60b9f0b75417c0d')
```

Block Diagram & Working:

Block Diagram:



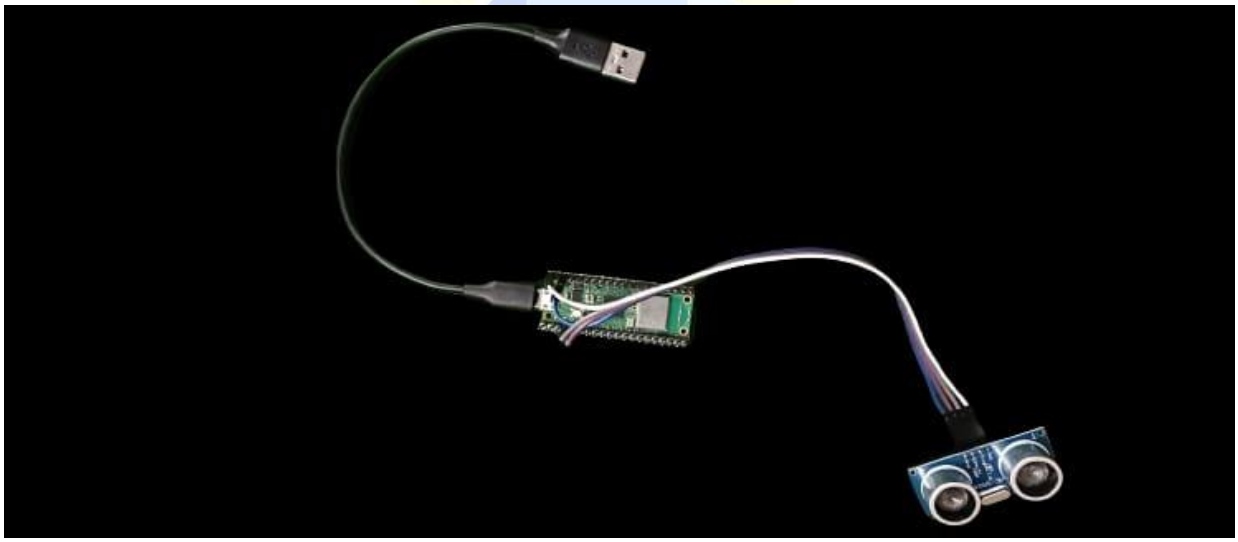
Working:

The project involves using an ultrasonic sensor connected to a Raspberry Pi Pico W to measure distance and sending an SMS message via the Twilio API if the measured distance exceeds a certain threshold.

The ultrasonic sensor sends out a high frequency sound wave, which bounces off an object and returns to the sensor. The time it takes for the sound wave to return is used to calculate the distance to the object.

The Raspberry Pi Pico W reads the distance data from the ultrasonic sensor and uses the urequests module to send an HTTP POST request to the Twilio API, which sends an SMS message to a specified phone number.

The network module is used to establish an internet connection, and the time module is used to introduce a delay between distance measurements.



Application:

- The application of this project could be in security systems, where an SMS alert can be sent when an object or a person enters a restricted area.

Result:

This project is that it provides a simple and inexpensive way to measure distance and send an SMS alert when an object is detected within a certain range. The Raspberry Pi Pico W and HC-SR04 ultrasonic sensor are both low-cost components that can be easily obtained and the Micropython programming language allows for easy programming and customization.