PRAC_6_B_LPHTUM003

Introduction

The system at hand is able to find the speed that a person waves at it. There are 2 inputs to the system a light sensor and the keyboard. The system works in the following way. A user can turn the system on and off through keyboard inputs. Once on the user can adjust the sensitivity off the system through keyboard inputs. The user can also wave at the system and if the speed of the users' wave is passes the systems current sensitivity setting the system will send the number of waves the person has made per second to a server.

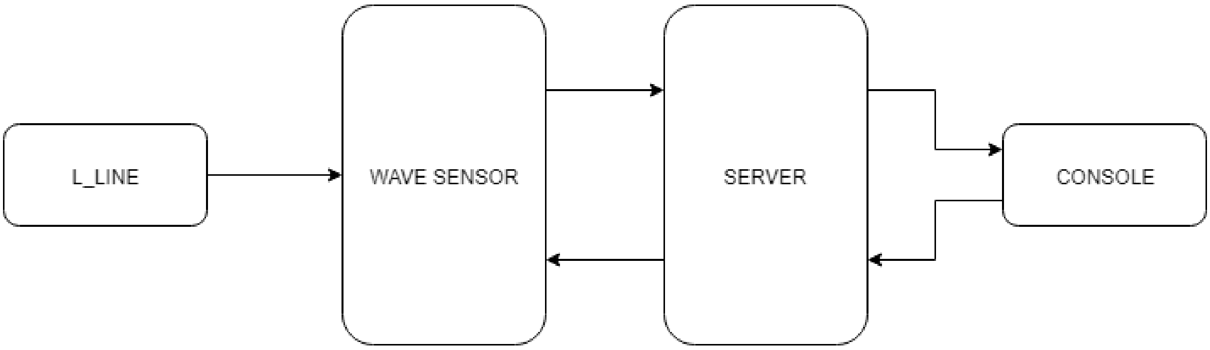

Figure 1 block diagram

Design and circuit

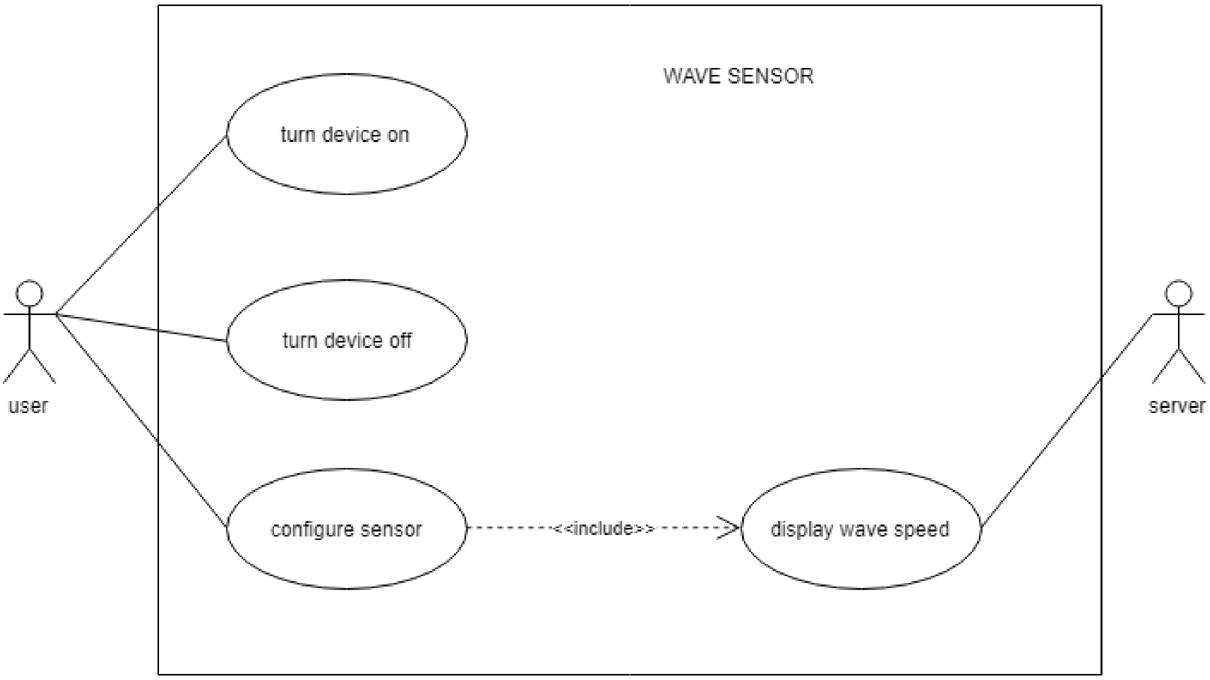

Figure 2 use case diagram

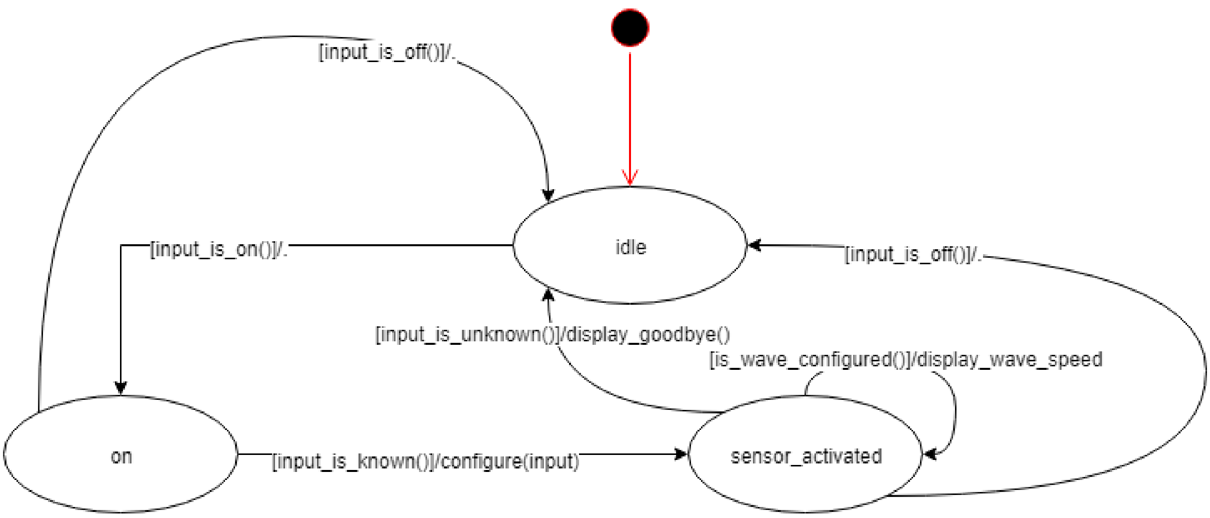

Figure 3 state chart diagram

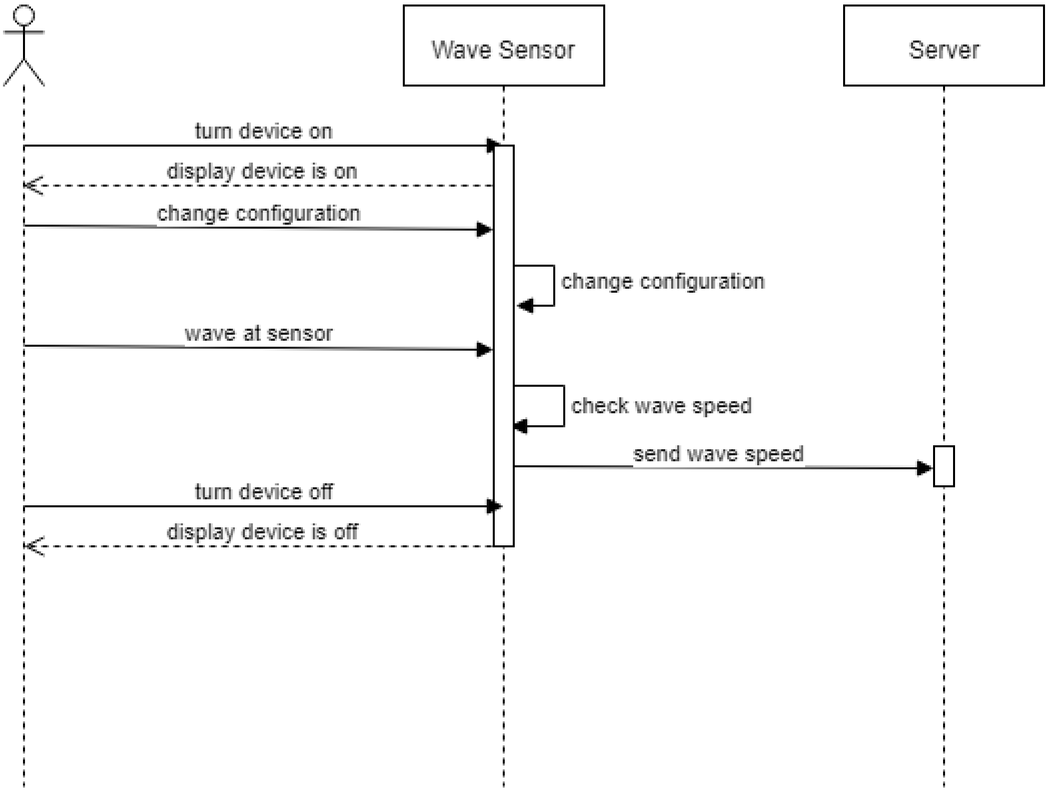

Figure 4 sequence diagram

Implementation and build process

Implementation

A single script called prac_6_b_lphtum003.py is able to read a configuration instruction sent to it by the user. Another functionality of the class is to set up a server where the number of waves per second are sent to the said server.

def set_config(config)

This function takes in configuration settings and instructs the wave sensor to respond appropriately.

- “W”: The system starts up and is reading in and sending the number of waves made by the user per seconds.
- “O”: The system shuts down
- “+”: Increments the required difference between light readings by 1
- “-“: Decrements the required difference between light readings by 1

```
def set_config(config):
    t1=time()
    while True:
        t2 = time()
        positive_count=1
        negative_count=1
        count = 0
        while config=="W":
            while t2 - t1 < 1:
                current_light = ldr.value*100
                sleep(0.1)
                next_light = ldr.value*100
                if next_light//1 > current_light//1:
                    count+=1
                t2 = time()
            speed_of_wave = count
            temp = t2
            t1 = temp
            print("%-3dwaves per second" % (speed_of_wave))
            count=0
        while config=="+":
            positive_count+=1
            negative_count=1
                        while t2 - t1 < 1:
                            current_light = ldr.value*100
                            sleep(0.1)
                            next_light = ldr.value*100
                            if next_light//1 > current_light//1+1*positive_count:
                                count+=1
                            t2 = time()
                        speed_of_wave = count
                        temp = t2
                        t1 = temp
                        print("%-3dwaves per second" % (speed_of_wave))
                        count=0

        while config=="-":
            negative_count+=1
            positive_count=1
                        while t2 - t1 < 1:
                            current_light = ldr.value*100
                            sleep(0.1)
                            next_light = ldr.value*100
                            if next_light//1 > current_light//1-1*negative_count:
                                count+=1
                            t2 = time()
                        speed_of_wave = count
                        temp = t2
                        t1 = temp
                        print("%-3dwaves per second" % (speed_of_wave))
                        count=0
```

Figure 5 wave sensor function

def handler(c, a)

Handler is a function that has 2 parameters a connection and address of an end system that is connected to the server. Handler takes in configuration settings sent to it by connected devices.

This function creates a wave sensor thread which has *def set_config(config)* as a runnable function and passes configuration settings to this thread and runs the wave thread which will give instructions to the wave sensor based on the settings passed.

```python
def handler(c, a):
        global connections
    global config
        help="Select one of the following commands:\n{\n\tW:ON,\n\tO:OFF,\n\t+:+100,\n\t-:-100\n}\n"

        while True:
                for connection in connections:
                        connection.send(help)

        data = c.recv(1024)

        if data[0]=="W":
            config="W"
                    for connection in connections:
                            connection.send("ON\n")
        elif data[0]=="O":
            for connection in connections:
                connection.send("OFF\n")
                c.close()
                break
        elif data[0]=="+":
                        config="+"
            for connection in connections:
                connection.send("+1\n")
        elif data[0]=="-":
                        config="-"
            for connection in connections:
                connection.send("-1\n")
        if not data:
                        connections.remove(c)
                        c.close()
            break
        wave_thread = threading.Thread(target=set_config, args=(config))
            wave_thread.start()
```

Figure 6 server function

Main loop

In this loop the server is waiting for devices to connect to it then these users will pass in configuration settings and run the wave sensor.

```python
while True:
    c, a = sock.accept()
    connection_thread = threading.Thread(target=handler, args=(c, a))
    connection_thread.daemon = True
    connection_thread.start()
    connections.append(c)
    print(c)
```

Figure 7 main function

Build process

The system was built upon the following components; RPi as an embedded system and A light sensor circuit

RPi embedded system

An RPi was used to model the server and main application that took in physical interrupts so as to create a cyber-physical system that implements IoT functionality by analysing these readings and sending them to a local server.

Light sensor circuit

This circuit is able to read in the change in light using an LDR. These inputs are sent to the RPi through appropriate gpio ports.

How to run the code

1. Run *prac_6_b_lphtum003.py*
2. Connect to the server running on another terminal using *telnet* on *localhost* at port *10000*
3. Start the wave sensor by typing *"W"*
4. Wave at the wave sensor
5. Go back to the terminal screen that ran the server
6. Watch the number of waves you made per second
7. Go back to the telnet terminal
8. Either increment ("+"), decrement ("-") or turn of the system ("O").

Testing

Using telnet to connect to the server

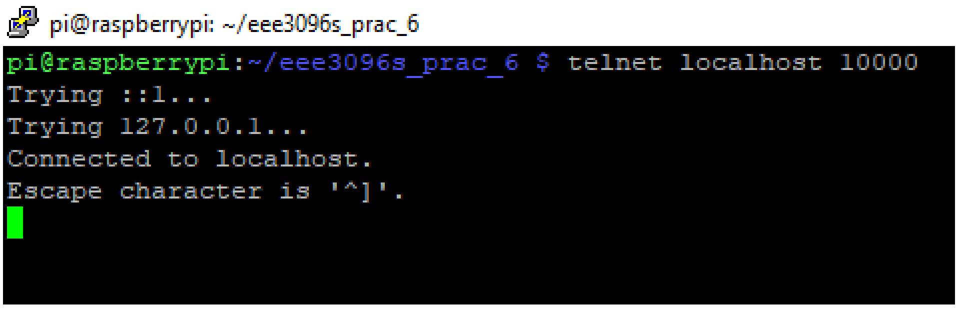

Figure 8 telnet connection

Testing server connection

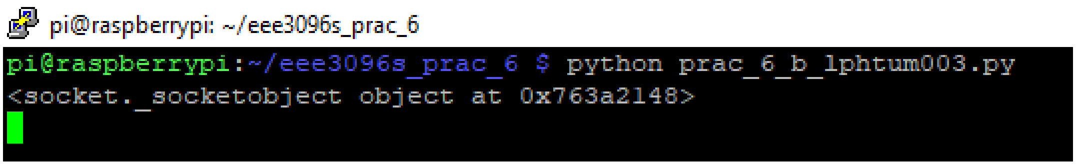

Figure 9 server connection

Entering all possible configuration settings

```
pi@raspberrypi: ~/eee3096s_prac_6
pi@raspberrypi:~/eee3096s_prac_6 $ telnet localhost 10000
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
W
ON
Select one of the following commands:
{
        W:ON,
        O:OFF,
        +:+100,
        -:-100
}
-
-1
Select one of the following commands:
{
        W:ON,
        O:OFF,
        +:+100,
        -:-100
}
+
+1
Select one of the following commands:
{
        W:ON,
        O:OFF,
        +:+100,
        -:-100
}
O
OFF
Connection closed by foreign host.
pi@raspberrypi:~/eee3096s_prac_6 $
```

Figure 10 user options

Display number of waves per second

```
2  waves per second
4  waves per second
2  waves per second
4  waves per second
6  waves per second
2  waves per second
5  waves per second
3  waves per second
```

Figure 11 server side messages

Conclusions

The system required the following functions

Table 1 requirements table

Requirements	Satisfied
User can turn system on	Yes
User can turn system off	Yes
System sends speed of waves to a server	Yes
User can change configuration settings	Yes

All requirements were met, and as such the project is completed successfully.