**DEPARTMENT OF COMPUTER ENGINEERING**

**SAN JOSE STATE UNIVERSITY**

**PROJECT REPORT**

*Presented to Department of Computer Engineering*

Course: CMPE 240-01

Semester: Spring 2017

**TOPIC: TALKING ALARM CLOCK USING RASPBERRY PI**



**PROJECT ADVISOR: JOHN VEIN**

**SUBMITTED BY**

**[Team 7]**

**AAJNA KARKI**

**ANIKET DALI**

**RAJEEV SAWANT**

**THRISHNA PALISSERY**

# Table of Contents

[Table of Contents 2](#_Toc482612569)

[1. Project Description 3](#_Toc482612570)

[2. Project Functionality 3](#_Toc482612571)

[3. Hardware Block Diagram 3](#_Toc482612572)

[4. Components 4](#_Toc482612573)

[ Datasheets of the Components 4](#_Toc482612574)

[ Schematic Diagram 5](#_Toc482612575)

[5. Flow Chart 6](#_Toc482612576)

[6. Test Plan 7](#_Toc482612577)

[7. Acknowledgement 20](#_Toc482612578)

[8. Conclusion 20](#_Toc482612579)

[9. Appendix 20](#_Toc482612580)

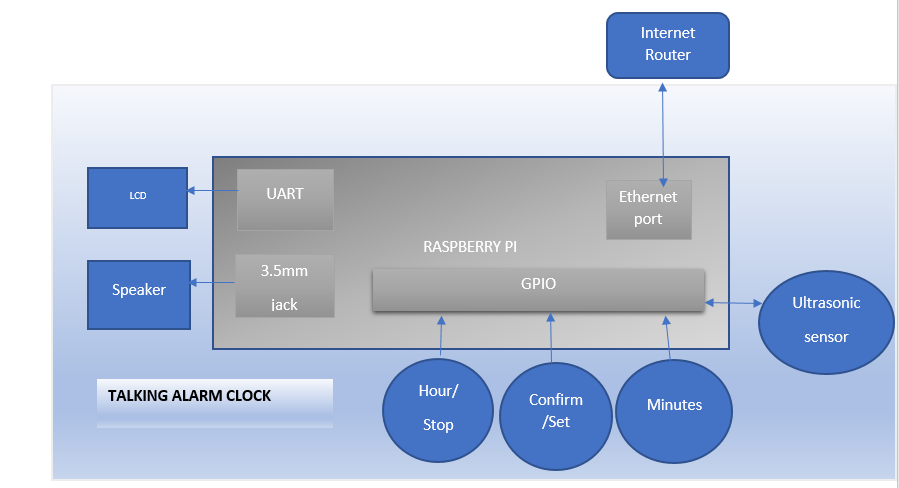
# Project Description

We propose to design a table alarm clock based on Raspberry pi 2-B. The alarm clock is specially designed to the meet the needs of visually impaired person. It will announce the current time on detection of presence of a person in the room. The clock will have switches to set the alarm. The switches will be configured with audio feedback to assist visually impaired. The clock will use internet connection to synchronize the time. The alarm clock will have a speaker interfaced through Raspberry Pi’s audio jack and LCD display to show the current time to make the clock versatile.

# Project Functionality

* Display current time.
* Speak out the current time on detection of human presence within 2 meters.
* Interface to set alarm with audio feedback to assist visually impaired person.
* Play music on alarm.
* Display alarm time.

# Hardware Block Diagram



**Figure 1: Hardware Block Diagram**

# Components

|  |  |  |
| --- | --- | --- |
| **Component** | **Source- Part number** | **Pricing (USD)** |
| Raspberry Pi 2 Model B | Digikey-1690-1005-ND | **43.75** |
| LCD Display | Digikey- NHD-0420D3Z-NSW-BBW-V3 | **32.00** |
| Push buttons (4) | Digikey part - EG-4791-ND | **0.53** |
| MaxBotix | Amazon - MB1010 | **31.07** |
| Speaker | Adafruit – 1363 | **9.95** |

**Table 1: Component Pricing**

## Datasheets of the Components

|  |  |
| --- | --- |
| * **Components** | **Datasheets** |
| Raspberry pi 2 model b |  |
| Maxbotix MB1010 |  |
| LCD Display |  |
| USB powered Speakers |  |

**Table 2: Datasheets of Component**

## Schematic Diagram

1. **Raspberry Pi – LCD Display**

c

**VCC(+5V) VCC(+5V)**

**RX TX**

**GND GND**

RASPBERRY PI

LCD

**Figure 2: MCU – LCD schematic diagram**

1. **Raspberry Pi -- Ultrasonic Sensor**

VCC

GND

GPIO 2

GPIO 3

VCC

GND

PWM

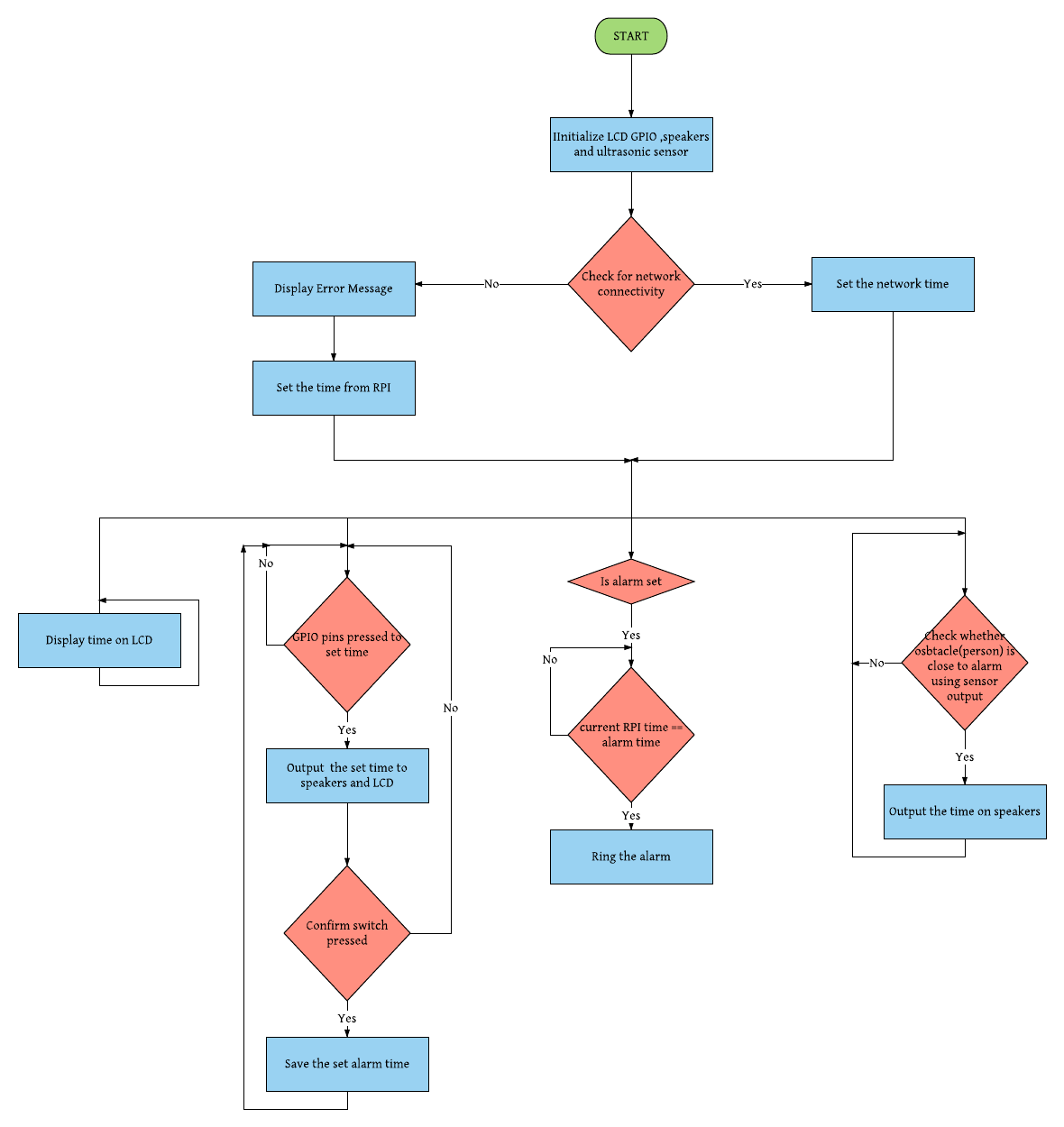
RX

Ultrasonic sensor

Raspberry Pi

**Figure 3: MCU – Sensor schematic diagram**

# Flow Chart



**Figure 4: Flow Chart**

# Test Plan

**Hardware – Raspberry Pi**

1. Test Name: - Raspberry Pi – Power on LED test
2. Test objective -

a1: To verify the controller board has adequate power.

1. Test procedure -

b1: Connect RPI to 5v supply and observe the power LED.

1. Test Vector –.

c1: 5V power supply.

1. Expected results –

d1: The power LED on RPI should glow else test fail.

1. Test Name: - Raspberry Pi – Ethernet interface

Test objective -

a1: To verify that controller boards Ethernet interface is working.

Test procedure -

b1: Power ON RPI, and plug in the Rj-45 cable to connect RPI to a working pre-tested network router.

Test Vector –.

c1: 5V power supply

c2: Ethernet cable

c3: Internet

Expected results –

d1: The ethernet Interface LEDs on the board should glow after 3 seconds.

1. Test Name: - Raspberry Pi- Audio jack

Test objective -

a1: To verify that controllers 3.5mm audio jack interface is working.

Test procedure -

b1: Power ON RPI, with the speakers connected to RPI’s 3.5 mm jack. RPI shall give out a beep upon initializing.

Test Vector –.

c1: 5V power supply

c2: Ethernet cable

c3: Internet

c4: Speakers

Expected results –

d1: RPI after boot up sequence gives a beep on the connected speaker.

1. Test Name: - Raspberry Pi – Power on LCD test

Test objective -

a1: To verify the LCD has adequate power on start.

Test procedure -

b1: Connect RPI to 5v supply.

Test Vector –.

c1: 5V power supply

c2: LCD

Expected results –

d1: The LCD display shall light up after RPI’s boot up sequence.

**Hardware – LCD Display**

1. Test Name: - LCD display – time parameters

Test objective -

a1: To verify that LCD displays the time in Hour : Minutes format

Test procedure -

b1: Power ON RPI, and plug in the Rj-45 cable to connect RPI to a working pre-tested network router. Wait for RPI to detect the time and display the current time on the LCD.

Test Vector –.

c1: 5V power supply.

c2: Ethernet cable

c3: Internet

c4: LCD

Expected results –

d1: The LCD shall display the current time in HR:MM format.

1. Test Name: - LCD display – Incrementing hours of alarm

Test objective -

a1: To verify that LCD increments the hours to set the alarm

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button ones.

b3: Wait for RPI’s audio feedback asking user to set the hours.

b4: Press the hour’s button to increment the hours to set the alarm.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

Expected results –

d1: The LCD shall display hours in an incrementing fashion on every successful press of hour button.

1. Test Name: - LCD display – Incrementing minutes of alarm

Test objective -

a1: To verify that LCD increments the minutes to set the alarm

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button ones.

b3: Wait for RPI’s audio feedback asking user to set the hours.

b4: Press the minute’s button to increment the minutes to set the alarm.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

Expected results –

d1: The LCD shall display minutes in an incrementing fashion on every successful press of minute button.

**Hardware – Speaker**

1. Test Name: - Raspberry Pi- Audio jack.

Test objective -

a1: To verify that controllers 3.5mm audio jack interface is working.

Test procedure -

b1: Power ON RPI, with the speakers connected to RPI’s 3.5 mm jack.

Test Vector –.

c1: 5V power supply

c2: speaker

Expected results –

d1: RPI after boot up sequence gives a beep on the connected speaker.

1. Test Name: - Speaker – Incrementing hours

Test objective -

a1: To verify that Speaker outputs the hour’s parameters while setting the hours.

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button ones.

b3: Wait for RPI’s audio feedback asking user to set the hours.

b4: Press the hour’s button to increment the hours to set the alarm.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The hour parameters outputted through speaker are loud and clear and they match with the hour’s parameters on LCD.

1. Test Name: - Speaker – Incrementing minutes

Test objective -

a1: To verify that Speaker outputs the minute’s parameters while setting the hours.

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button ones.

b3: Wait for RPI’s audio feedback asking user to set the minutes.

b4: Press the minute’s button to increment the minutes to set the alarm.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The LCD shall display minutes in an incrementing fashion on every successful press of minute button.

1. Test Name: - Speaker – user instruction to set alarm - Hours

Test objective -

a1: To verify that speaker outputs the instructions to configure the hours to set the alarm.

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button ones.

b3: Wait for RPI’s audio feedback asking user to set the hours.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The instruction outputted through speaker to set hours is loud and clear.

1. Test Name: - Speaker – user instruction to set alarm - Minutes

Test objective -

a1: To verify that speaker outputs the instructions to configure the hours to set the alarm.

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button twice.

b3: Wait for RPI’s audio feedback to notify user on the alarm time.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The instruction outputted through speaker to set minutes is loud and clear.

1. Test Name: - Speaker – user notification of the alarm set.

Test objective -

a1: To verify that speaker outputs the instructions to notify user the time when the next alarm is set.

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Configure the alarm and press confirm.

b3: Wait for RPI’s audio feedback asking user to set the minutes.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The instruction outputted through speaker on alarm time is loud and clear.

1. Test Name: - Speaker – current time.

Test objective -

a1: To verify that speaker outputs the current time on detection of object nearby

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Place an object at 1 meter from RPI

b3: wait for system to output the current time.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The current time outputted through speaker is loud and clear.

1. Test Name: - Speaker – Alarm tone

Test objective -

a1: To verify that speaker outputs alarm tone once the alarm goes off.

Test procedure -

b1: configure the alarm for 1 minute duration from current time

b2: Press the set/confirm button twice.

b3: wait for alarm to go off.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The speaker shall output the alarm ringtone loud and clearly.

**Hardware – Button**

1. Test Name: - Buttons – Hour button -Hr.

Test objective -

a1: To verify that Hour’s button scrolls the hours to set the alarm

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button ones.

b3: Wait for RPI’s audio feedback asking user to set the hours.

b4: Press the hour’s button to increment the hours to set the alarm.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The tactile button feedback on press was adequate to understand the button press.

d2: single button press was registered only once.

d3: number of hours incremented by 1.

1. Test Name: - Buttons – Hour button - Min

Test objective -

a1: To verify that Hour’s button scrolls the minutes to set the alarm

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button twice.

b3: Wait for RPI’s audio feedback asking user to set the minutes.

b4: Press the hour’s button to increment the minutes by 10 to set the alarm.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The tactile button feedback on press was adequate to understand the button press.

d2: single button press was registered only once.

d3: number of minutes incremented by 10.

1. Test Name: - Buttons – Minute button.

Test objective -

a1: To verify that Minute’s button scrolls the minutes to set the alarm

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button ones.

b3: Wait for RPI’s audio feedback asking user to set the minutes.

b4: Press the minute’s button to increment the minutes to set the alarm.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The tactile button feedback on press was adequate to understand the button press.

d2: single button press was registered only once.

d3: number of minutes incremented by 1.

1. Test Name: - Buttons – set/confirm - to enable alarm setting

Test objective -

a1: To verify that set /confirm button enables the alarm setting.

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button once.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The tactile button feedback on press was adequate to understand the button press.

d2: single button press was registered only once.

d3: Speaker plays the instruction only to set alarm hours.

1. Test Name: - Buttons – set/confirm - to confirm hour

Test objective -

a1: To verify that set /confirm button confirms hour set .

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button once after selecting the hour.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The tactile button feedback on press was adequate to understand the button press.

d2: single button press was registered only once.

d3: Speaker plays the instruction only to set minutes.

1. Test Name: - Buttons – set/confirm – confirm alarm time.

Test objective -

a1: To verify that set /confirm button confirms set alarm time.

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button once.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The tactile button feedback on press was adequate to understand the button press.

d2: Speaker plays the confirmed alarm time.

1. Test Name: - Buttons – hour - to stop alarm

Test objective -

a1: To verify that pressing hour button stops the alarm.

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Press the set/confirm button ones.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: The tactile button feedback on press was adequate to understand the button press.

d2: single button press was registered only once.

d3: Speaker stops playing alarm sound.

**Hardware – Ultrasonic range detector sensor**

1. Test Name: - Obstacle detection

Test objective -

a1: To verify that range detection sensor detects the presence of person nearby.

Test procedure -

b1: Allow RPI to boot up and display time.

b2: Place an object at 1 meter away from the clock in line of sight of the sensor.

Test Vector –

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: Observe that Alarm clock shall output the current time only once.

**Software– Talking Alarm Clock**

1. Test Name: - Detection of local time on boot up

Test objective -

a1: To verify that talking alarm clock retrieves the current local time through internet.

Test procedure -

b1: Plug Talking alarm clock to pre- tested network router connected to internet through pre-verified ethernet cable via ethernet interface.

b2: power up.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

c5: Internet

Expected results –

d1: Talking alarm shall display the current local time on initialization.

1. Test Name: - Detection of no internet connectivity

Test objective -

a1: To verify that Talking alarm can detect no internet connection on startup.

a2: and display an appropriate message to user and proceed to normal behavior.

Test procedure -

b1: Power up Talking alarm clock without connecting to internet

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: Talking alarm shall display no internet connectivity error after 20 seconds after bootup.

d2: It should display ‘Network Error!’ on lcd.

1. Test Name: - Detection of human presence

Test objective -

a1: To verify that Talking alarm clock outputs the current time on detection of human presence in 1 meter distance from the viewing side of the clock.

Test procedure -

b1: Power on Talking alarm clock.

b2: Place an object a meter away from Talking alarm’s display side.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

Expected results –

d1: Talking alarm shall speak output the current time through the attached speaker.

1. Test Name: - Display current time.

Test objective -

a1: To verify that Talking alarm clock displays the current time.

Test procedure -

b1: Connect talking alarm clock to internet through verified connection and power on talking alarm clock.

b2: Wait for it to display the current local time

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c4: speaker

c5: internet

Expected results –

d1: Talking alarm clock should display the current local time.

1. Test Name: - Rollover hours - Minutes

Test objective -

a1: To verify that Talking alarm clock rolls over from 59-00 for minutes

Test procedure -

b1: Configure the software to start with the local time 23:59.

b2: Do not connect Talking alarm clock to internet.

b3: Wait for 1 minute.

b4: observe the display’s output.

Test Vector –.

c1: 5V power supply

c2: LCD

c3: push button/buttons

c

Expected results –

d1: The new time displayed by talking alarm clock shall be 0.00 after roll over.

# Acknowledgement

We would like to extend our sincere gratitude to Professor John Vein for his guidance and constant supervision for completing our project on time.

# Conclusion

It was a good learning experience for us while working on this project. This project took us through the various phases of project development and gave us a real insight into the world of Software/Firmware Engineering.

# Appendix

1. **REFERENCES**

<http://www.raspberry-projects.com/pi/programming-in-c/uart-serial-port/using-the-uart>.

<http://elinux.org/RPi_GPIO_Code_Samples>

<http://www.maxbotix.com/documents/LV-MaxSonar-EZ_Datasheet.pdf>

1. **Source Code**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h> // Used for UART

#include <fcntl.h> // Used for UART

#include <termios.h> // Used for UART

#include <sys/time.h> // Used for sys time

#include <wiringPi.h> // Used for UART and GPIO

#include <errno.h> // error messages

#include <pthread.h> // multi threading

/\* uart fd \*/

int uart0\_filestream = -1;

pthread\_t thread\_lcd;

// forward declarations

void initialize\_uart();

void\* display();

void send\_commands\_lcd(char\* cmd,int size);

void display\_text(char\* text,int size);

void display\_alarm();

void display\_time();

/\* sensor \*/

pthread\_t thread\_sensor;

#define PIN\_PW 2

#define PIN\_RX 3

static volatile int state;

struct timeval now\_rising;

int sensor\_flag = 0;

int sensor\_output = 0;

int alarm\_clear = 0;

static volatile int network\_down = 0;

int trigger\_flag = 1;

/\* GPIO \*/

pthread\_t thread\_gpio;

pthread\_t thread\_sound;

pthread\_t thread\_alarm;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* defines section \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define BUTTON\_CONFIRM 4

#define BUTTON\_HOUR 0

#define BUTTON\_MIN 5

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* global variable section \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// button pressed - 1 when any othe button is pressed (hr, min or set/confirm)

volatile int button\_pressed = 0;

// button number to identify the button pressed.

// 1 - hr

// 2 - min

// 3 - set/confirm

volatile int button\_number = 0;

// variable to store hr

static volatile int hr = 0;

// variable to store min

static volatile int min = 0;

// variable to set hr set min and confirm.

// 1 - set hr

// 2 - set min and set alarm

// 0 - alarm is set

static volatile int confirm = 0;

volatile int gpio\_set = 0;

volatile int alarm\_set = 0;

// array for hr values

char cmd\_hr [ ][30] = {

//0 1 2 3 4 5 6

{"echo 0 | festival --tts "},{ "echo 1 | festival --tts "}, {"echo 2 | festival --tts "},{ "echo 3 | festival --tts "},{ "echo 4 | festival --tts "}, {"echo 5 | festival --tts "},{ "echo 6 | festival --tts "},

//7 8 9 10 11 12 13

{"echo 7 | festival --tts "},{ "echo 8 | festival --tts "}, {"echo 9 | festival --tts "},{ "echo 10 | festival --tts "},{ "echo 11 | festival --tts "}, {"echo 12 | festival --tts "},{ "echo 13 | festival --tts "},

//14 15 16 17 18 19 20

{"echo 14 | festival --tts "},{ "echo 15 | festival --tts "}, {"echo 16 | festival --tts "},{ "echo 17 | festival --tts "},{ "echo 18 | festival --tts "}, {"echo 19 | festival --tts "},{ "echo 20 | festival --tts "},

// 21 22 23 24

{"echo 21 | festival --tts "},{ "echo 22 | festival --tts "}, {"echo 23 | festival --tts "},{ "echo 24 | festival --tts "} };

// array for minutes values

char cmd\_min [ ][60] = {

//0 1 2 3 4 5 6

{"echo 0 | festival --tts "},{ "echo 1 | festival --tts "}, {"echo 2 | festival --tts "},{ "echo 3 | festival --tts "},{ "echo 4 | festival --tts "}, {"echo 5 | festival --tts "},{ "echo 6 | festival --tts "},

//7 8 9 10 11 12 13

{"echo 7 | festival --tts "},{ "echo 8 | festival --tts "}, {"echo 9 | festival --tts "},{ "echo 10 | festival --tts "},{ "echo 11 | festival --tts "}, {"echo 12 | festival --tts "},{ "echo 13 | festival --tts "},

//14 15 16 17 18 19 20

{"echo 14 | festival --tts "},{ "echo 15 | festival --tts "}, {"echo 16 | festival --tts "},{ "echo 17 | festival --tts "},{ "echo 18 | festival --tts "}, {"echo 19 | festival --tts "},{ "echo 20 | festival --tts "},

// 21 22 23 24 25 26 27

{"echo 21 | festival --tts "},{ "echo 22 | festival --tts "}, {"echo 23 | festival --tts "},{ "echo 24 | festival --tts "},{"echo 25 | festival --tts "},{ "echo 26 | festival --tts "},{ "echo 27 | festival --tts "},

//28 29 30 31 32 33 34

{"echo 28 | festival --tts "},{ "echo 29 | festival --tts "}, {"echo 30 | festival --tts "},{ "echo 31 | festival --tts "},{ "echo 32 | festival --tts "}, {"echo 33 | festival --tts "},{ "echo 34 | festival --tts "},

//35 36 37 38 39 40 41

{"echo 35 | festival --tts "},{ "echo 36 | festival --tts "}, {"echo 37 | festival --tts "},{ "echo 38 | festival --tts "},{ "echo 39 | festival --tts "}, {"echo 40 | festival --tts "},{ "echo 41 | festival --tts "},

// 42 43 44 45 46 47 48

{"echo 42 | festival --tts "},{ "echo 43 | festival --tts "}, {"echo 44 | festival --tts "},{ "echo 45 | festival --tts "},{"echo 46 | festival --tts "},{ "echo 47 | festival --tts "},{ "echo 48 | festival --tts "},

//49 50 51 52 53 54 55

{"echo 49 | festival --tts "},{ "echo 50 | festival --tts "}, {"echo 51 | festival --tts "},{ "echo 52 | festival --tts "},{ "echo 53 | festival --tts "}, {"echo 54 | festival --tts "},{ "echo 55 | festival --tts "},

//56 57 58 59

{"echo 56 | festival --tts "},{ "echo 57 | festival --tts "}, {"echo 58 | festival --tts "},{ "echo 59 | festival --tts "}};

// instructions to guide through the process

char welcome[50] = "echo Please set alarm | festival --tts ";

char instr1[150] = "echo Please, use the hour button, to set hours in 24 hours format and then press, confrm button, to confirm| festival --tts ";

char instr2[150] = "echo Please, use the minute button, to set minutes, and then press, confrm button, to set alarm | festival --tts ";

char set[50] = "echo Alarm is set for | festival --tts ";

char hours[40] = "echo hours and | festival --tts ";

char minutes[35] = "echo minutes | festival --tts ";

char wakeup[35] = "echo wake up | festival --tts ";

char network\_error[45] = "echo Network is down | festival --tts ";

void uart\_init()

{

uart0\_filestream = open("/dev/ttyAMA0", O\_RDWR | O\_NOCTTY | O\_NDELAY); //Open in non blocking read/write mode

if (uart0\_filestream == -1)

{

//ERROR - CAN'T OPEN SERIAL PORT

printf("Error - Unable to open UART. Ensure it is not in use by another application\n");

}

struct termios options;

tcgetattr(uart0\_filestream, &options);

options.c\_cflag = B9600 | CS8 | CLOCAL | CREAD; //<Set baud rate

options.c\_iflag = IGNPAR;

options.c\_oflag = 0;

options.c\_lflag = 0;

tcflush(uart0\_filestream, TCIFLUSH);

tcsetattr(uart0\_filestream, TCSANOW, &options);

}

void \*display\_lcd()

{

while(1)

{

if(network\_down == 1)

{

char clrLine[2] = {0xFE,0x51};

send\_commands\_lcd(clrLine,2);

char setLine[3] = {0xFE,0x45,0x40};

send\_commands\_lcd(setLine,3);

char error[15] = "Network Error!";

display\_text(error,14);

}

else if(gpio\_set != 1)

{

if(alarm\_set!=1)

{

display\_time();

}

else

{

display\_time();

display\_alarm();

}

}

else

{

char clearLCD[2]= {0xFE,0x51};

send\_commands\_lcd(clearLCD,2);

char setLine[3] = {0xFE,0x45,0x40};

send\_commands\_lcd(setLine,3);

display\_alarm();

}

delay (500);

}

}

void \*check\_alarm()

{

time\_t rawtime;

struct tm \* timeinfo;

for(;;)

{

time ( &rawtime );

timeinfo = localtime ( &rawtime );

if(timeinfo->tm\_hour == hr && timeinfo->tm\_min == min && alarm\_set == 1)

{

system(wakeup);

alarm\_set = 0;

alarm\_clear = 0;

while(!alarm\_clear)

system(wakeup);

}

}

}

void send\_commands\_lcd(char\* cmd,int size)

{

int j=0;

write(uart0\_filestream,cmd,size);

delay (5);

}

void display\_text(char\* text,int size)

{

int count = write(uart0\_filestream,text,size);

if (count < 0)

{

printf("UART TX error\n");

}

}

void display\_time()

{

int size = 0;

char clearLCD[2]= {0xFE,0x51};

send\_commands\_lcd(clearLCD,2);

char setLine[3] = {0xFE,0x45,0x00};

send\_commands\_lcd(setLine,3);

char text[10] = "Time: ";

size = strlen(text);

display\_text(text,size);

/\*Displays time\*/

time\_t rawtime;

struct tm \* timeinfo;

time ( &rawtime );

timeinfo = localtime ( &rawtime );

size = strftime (text,80, "%H:%M", timeinfo);

display\_text(text,size);

}

void display\_alarm()

{

int size = 0;

char text[10]= {0};

char setLine[3] = {0xFE,0x45,0x54};

send\_commands\_lcd(setLine,3);

strncpy(text,"Alarm: ",7);

size = strlen(text);

display\_text(text,size);

size = sprintf(text,"%d",hr);

display\_text(text,size);

strncpy(text,":",1);

size = strlen(text);

display\_text(text,size);

size = sprintf(text,"%d", min);

display\_text(text,size);

}

void start()

{

if(state == 0)

{

gettimeofday(&now\_rising, NULL);

// printf("Rising\n");

state = 1;

}else

{

struct timeval now\_falling;

gettimeofday(&now\_falling, NULL);

int distance = ((now\_falling.tv\_usec-now\_rising.tv\_usec)/147);

//printf("Distance :: %d\n",distance);

if((distance > 6) && distance < 20 )

{

printf("S");sensor\_flag = 1;

}

else

{

printf("T");trigger\_flag = 1;

}

state = 0;

}

}

void sensor\_init()

{

wiringPiSetup();

// Set pin to output in case it's not

pinMode(PIN\_PW, INPUT);

pinMode(PIN\_RX, OUTPUT);

wiringPiISR(PIN\_PW, INT\_EDGE\_BOTH, &start);

}

void \*sensor\_trigger()

{

for (;;)

{

digitalWrite(PIN\_RX,1);

delay(49);

digitalWrite(PIN\_RX,0);

sleep(1);

}

}

// interrupt routine for Hours

void myInterruptHr(void)

{

button\_pressed = 1;

button\_number = 1;

}

// Interrupt routine for minutes

void myInterruptMin(void)

{

button\_pressed = 1;

button\_number = 2;

}

// Interrupt routine for set/confirm

void myInterruptConfirm(void)

{

button\_pressed = 1;

button\_number = 3;

}

void gpio\_init()

{

// pin mode

pinMode (BUTTON\_HOUR,INPUT);

pinMode (BUTTON\_MIN,INPUT);

pinMode (BUTTON\_CONFIRM,INPUT);

// register interrupts

wiringPiISR(BUTTON\_HOUR,INT\_EDGE\_RISING,&myInterruptHr);

wiringPiISR(BUTTON\_MIN,INT\_EDGE\_RISING,&myInterruptMin);

wiringPiISR(BUTTON\_CONFIRM,INT\_EDGE\_RISING,&myInterruptConfirm);

}

void \*gpio\_read()

{

for (;;)

{

// printf("gpio\_set:%d \n",gpio\_set);

// printf("alarm set:%d \n",alarm\_set);

//printf("#");

if (button\_pressed == 1)

{

switch(button\_number)

{

// hour button pressed

case 1: if(HIGH == digitalRead(BUTTON\_HOUR))

{

// debounce delay - may vary

delay (250);

printf("hr:%d , min:%d\n",hr,min);

// if we are in state to set hours then set hours

alarm\_clear = 1;

if( confirm == 1)

{

hr++;

system(cmd\_hr[hr]);

if(hr>23)

{

hr =0;

}

}

// if we are in state to set minutes then set minutes by incrementing by 10

else if( confirm == 2)

{

min = min+10;

if(min>59)

{

min = min%60;

}

system(cmd\_min[min]);

}

}

// false click - relax!!

else

{

printf("just a debounce hr\n");

}

break;

// set minutes

case 2: if(HIGH == digitalRead(BUTTON\_MIN))

{

// debounce delay

delay (250);

printf("hr:%d , min:%d\n",hr,min);

// if we are in state to set minutes then only carry on

if( confirm == 2)

{

min++;

system(cmd\_min[min]);

if(min>59)

{

min =0;

}

}

}

else

{

printf("just a debounce min \n");

}

break;

// button is pressed to set hr or set min-alarm

case 3: if(HIGH == digitalRead(BUTTON\_CONFIRM))

{

delay (250);

// alarm set process begins ask user to provide hr info

if (confirm == 0)

{

while(sensor\_output ==1);

// user has entered settings

gpio\_set = 1;

// intimate that he has opted to set alarm

system(welcome);

// ask him to use hour button and then confirm it

system(instr1);

hr = 0;

min = 0;

confirm++;

}

// user has confirmed hours and now ask him to set minutes - set alarm

else if(confirm == 1)

{

system(instr2);

confirm++;

}

else if (confirm == 2)

{

confirm = 0;

// user has set the alarm

alarm\_set = 1;

system(set);

system(cmd\_hr[hr]);

system(hours);

system(cmd\_min[min]);

system(minutes);

// user has exited settings

gpio\_set = 0;

}

}

// frankie says no

else

{

printf("just a debounce conf\n");

}

break;

default: break;

}

// clear the flag

button\_pressed = 0;

}

}

}

void \*check\_obstacle()

{

time\_t rawtime;

struct tm \* timeinfo;

for(;;)

{

time ( &rawtime );

timeinfo = localtime ( &rawtime );

if(sensor\_flag == 1 && trigger\_flag != 0)

{

printf("X %d %d\n",timeinfo->tm\_hour,timeinfo->tm\_min); // system(cmd\_hr[0]);

if(gpio\_set != 1)

{

sensor\_output = 1;

system(cmd\_hr[timeinfo->tm\_hour]);

system(hours);

system(cmd\_min[timeinfo->tm\_min]);

system(minutes);

sensor\_output = 0;

}

sensor\_flag = 0;

trigger\_flag = 0;

// sleep(100);

}

}

}

int main()

{

FILE \*p = fopen("/home/pi/error.txt","r");

if(p !=NULL)

{

network\_down = 1;

system(network\_error);

uart\_init();

pthread\_create(&thread\_lcd,NULL,display\_lcd,NULL);

}

else

{ /\* UART INIT \*/

uart\_init();

/\* SENSOR INIT \*/

sensor\_init();

/\* GPIO INIT \*/

gpio\_init();

/\* GPIO READ \*/

pthread\_create(&thread\_sensor,NULL,sensor\_trigger,NULL);

/\* LCD DISPLAY \*/

pthread\_create(&thread\_lcd,NULL,display\_lcd,NULL);

/\* SENSOR TRIGEER \*/

pthread\_create(&thread\_gpio,NULL,gpio\_read,NULL);

pthread\_create(&thread\_sound,NULL,check\_obstacle,NULL);

pthread\_create(&thread\_alarm,NULL,check\_alarm,NULL);

} while(1);

printf(" Program Ending \n");

return 0;

}

**III . Source Coude Packages Installed**

WiringPi GPIO Library

Festival Package for Text to Speech Synthesis