Microprocessor and Interfacing

J-COMPONENT REPORT

Smart Mirror as Notice Board

Professor: **Arvind Kumar**

Course code : CSE2006
Slot: F1

Submitted by:

Jay agarwal	19BCE0848
Ayush bansal	19BCE0862
Sarthak bhardwaj	19BCE0859

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Table of contents

TOPIC	PAGE
1. Abstract	4
2. Introduction	5
2.1. Objective	5
2.2. Application	5
3. Literature Survey	6
4. System Analysis	8
4.1. Existing Systems	8
4.2. Proposed System	8
5. Requirement Specification	9
5.1. Hardware Requirements	9
5.2. Software Requirements	9
5.3. Architecture Diagram	9
5.4. Circuit Diagram	10
5.5. Pin Diagram	10
6. Implementation	12
6.1. PIR Sensor	12
6.2. 16 x 2 LCD Display	12
7. Conclusion	13
8. Appendices	13
9. References	21

1. Abstract

Notice Board is essential to display in any organization or at public places to provide the information. In the present situation, the notice/advertisement boards are almost always managed manually. But pasting various notices on a daily basis is a difficult/long process. This wastes a lot of time, paper, and manpower. Android notice boards are developed recently but still, it is found that there is still ignorance is reading notice board. So to make people read the notices we use the mirror display. We present the development of an innovating appliance that incorporates interactive services of information, erred through a user interface on the surface of a mirror.

Our work is based on the idea that we all look in the mirror when we go out, so why wouldn't the mirror become smart. In this system, we propose a remotely send notice to smart mirror from an Android application based on Raspberry pi card. Now the world is moving towards automation, so in this world, if we want to do some changes in the previously used system, we have to use the new techniques. The wireless operation provides fast transmission over long-range communication. It saves resources and time. Data can be sent from the remote location. User authentication is provided.

2. Introduction

We propose an Ambient Tablet for a Smart notice board system using the concept of Internet of Things. The system uses Raspberry pi- Single Board Computer (SBC), acting as the processor with Wi-Fi network connectivity. It comprises of wireless sensor nodes, sensor modules and a display unit forming the additional requirements of the system. The proposed system uses Evolutionary Prototyping methodology. The prime feature of the system is to display updates seamlessly to the users. The system's root user updates the notices that are to be displayed on the designed interface. The system enables the users to use voice recognition tool to interact with the notice board. It has the ability to monitor the surrounding, by availing the sensor module embedded to the framework. It also has an additional ability to automate the locality with ministration of sensors. The system is coded using Python for Rpi environment. The key motive of this IoT based Ambient Reflector, is to unveil the notices promptly. The proposed system replaces printed notices which saves time, shrinks man power and assists in achieving a greener environment.

2.1. Objective –

Our main objective is to conceptualize, design and develop a mirror that acts as a smart notice board to the user providing automation of surroundings and a connected sensor module to seek the surveillance of the surroundings, and allow user to execute voice-based commands.

2.2. Applications –

The main goal of this system is to develop a wireless notice board that display message sent from the user and to design a simple, easy to install, user friendly system, which can receive and display notice in a particular manner with respect to date and time which will help the user to easily keep the track of notice board every day and each time he uses the system. Wi-Fi is the wireless technology used.

3. Literature Survey

S.NO	Paper Title	Name of Paper, year	Review
1	**Diy Smart Mirror**	**Springer International Publishing ,2018. Belma Ramic-Brkic ,Sadeta Kulovic, (2018)**	**IoT is an interconnection of Wireless Sensor Network (WSN) devices which includes embedded devices with wireless sensors. Scrolling display board is a common sight today. Advertisement is going digital. The proposed Ambient Reflector is an IOT enabled Smart Notice Board that displays the notices on the reflector, which is easy to update.**
2	**Fitmirror: A Smart Mirror For Positive Affect In Everyday User Morning Routines**	**Acm Publication , November 2016. Alexander Nikic Frank Honold,Daniel Besserer, Felix Schüssel , Johannes Bäurle**	**The goal of Smart Mirror is to positively affect the user ' s feelings by increasing his/her motivation, mood and feeling of fitness. Smart mirrors make use of two way reflectors which acts as both a mirror and an interface to trace relevant information such as time and date, weather**
3	**Smart Mirror For Smart Life**	**Ieee Publication,March 2017. Kamaruzzaman Jahidin , Muhammad Mu'izzudeenYusri , Mohammad Syafwan Arshad, Rohayanti Hassan , Shahreen Kasim , Zubaile Abdullah Husni Ruslai (2017)**	**A Digital Notice Board based on GSM operated at the Universities, colleges, schools, hospitals, railway stations, gardens etc. for exhibiting day-to-day information continuously or at regular intervals without affecting the surrounding environment. It offers pliability to display flash notices, information or announcements faster than the programmable system**

4	**“Smart Mirror:A Reflective Interface To Maximize Producticity**	**International Journal Of Computer Apllications,2017. Maninder Jeet Kaur, Piyush Maheshwari,SarthakAnand, (May 2017)**	**The existing systems that have been built in past by various researchers have successfully displayed notices but they have certain issues to be solved. The proposed system solves some of the major issues such as updating notices anywhere within the network, increase in ease of readability and use of voice commands which is used for interaction.**
5	**IOT BASED SMART MIRROR WITH NEWS AND TEMPERATURE**	**1Apurva Joshi, *1Prerana Shukla, *1Sanya Verma, *1Srishti Shakti**	**Their proposed system allows them to build such mirrors that allow for mirrors to receive news online and display it on the mirror screen along with other details including current temperature from an open weather platform for a futuristic and modern lifestyle. Their system uses a raspberry pi based processor board along with displays that are interfaced together. They use a precisely modelled panel to construct the outer frame**
6	**Future IoT based on Smart Mirror: A Literature Review**	**Seto Benson Handoyo, Michael Vincentius Setiawan , Mikhael Valensius and Mochammad Haldi Widianto**	**Smart Mirrors and IoT can also help in people ' s daily lives such as can be used for smart parking. In implementing IoT in the future, the challenges that will be faced are very diverse, such as the accuracy of analysis, data security and so on. These challenges must be prevented and their impact minimized so that IoT can be of maximum use in the future**

4. System Analysis

4.1. Existing Systems –

The existing systems that have been built in past by various researchers have successfully displayed notices but they have certain issues to be solved. The earlier designed LED based Scrolling Notice Board [5] made use of LED matrix-based display system. The intelligence and control were done using ATMEGA 16 microcontroller to display the message. The main issue faced here includes difficulty in reading the scrolling message

4.2. Proposed System –

Now the world is moving towards automation, so in this world, if we want to do some changes in the previously used system, we have to use the new techniques. The wireless operation provides fast transmission over long-range communication. It saves resources and time. Data can be sent from the remote location. User authentication is provided. Our system uses Raspberry pi- Single Board Computer (SBC), acting as the processor with Wi-Fi network connectivity. It comprises of wireless sensor nodes, sensor modules and a display unit forming the additional requirements of the system.

5. Requirement Specification

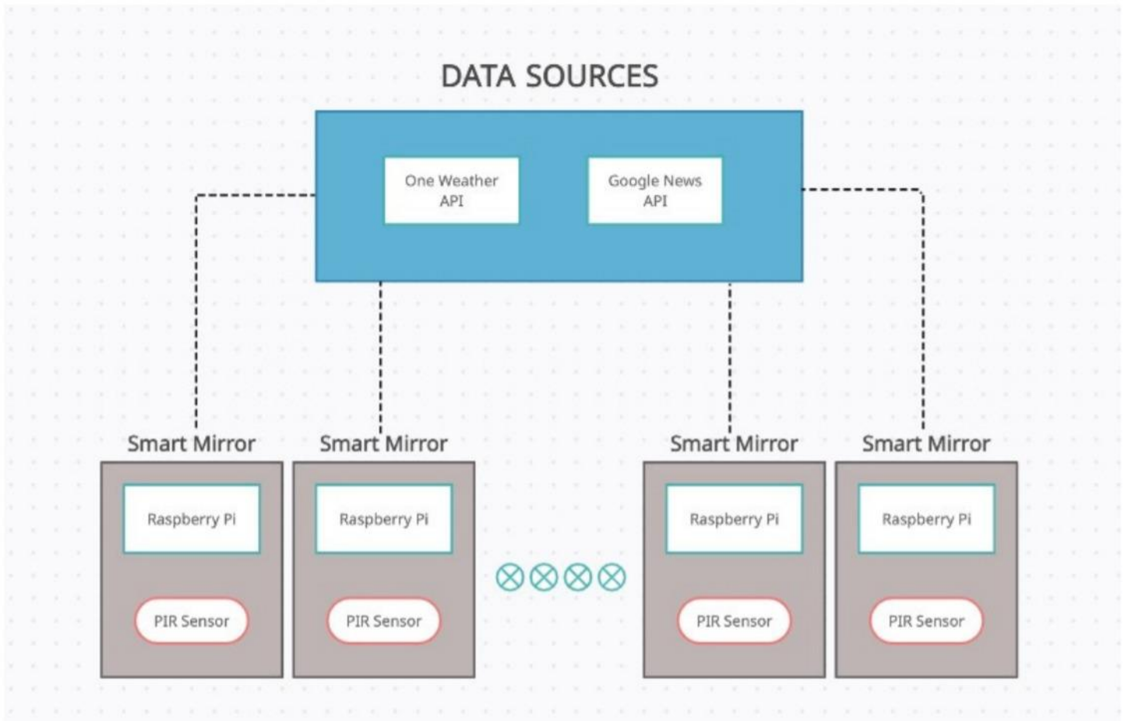
5.1. Hardware Requirements –

- A PC/Laptop with strong processor and GPU
- Raspberry Pi
- IR Proximity Sensor
- Two – way mirror

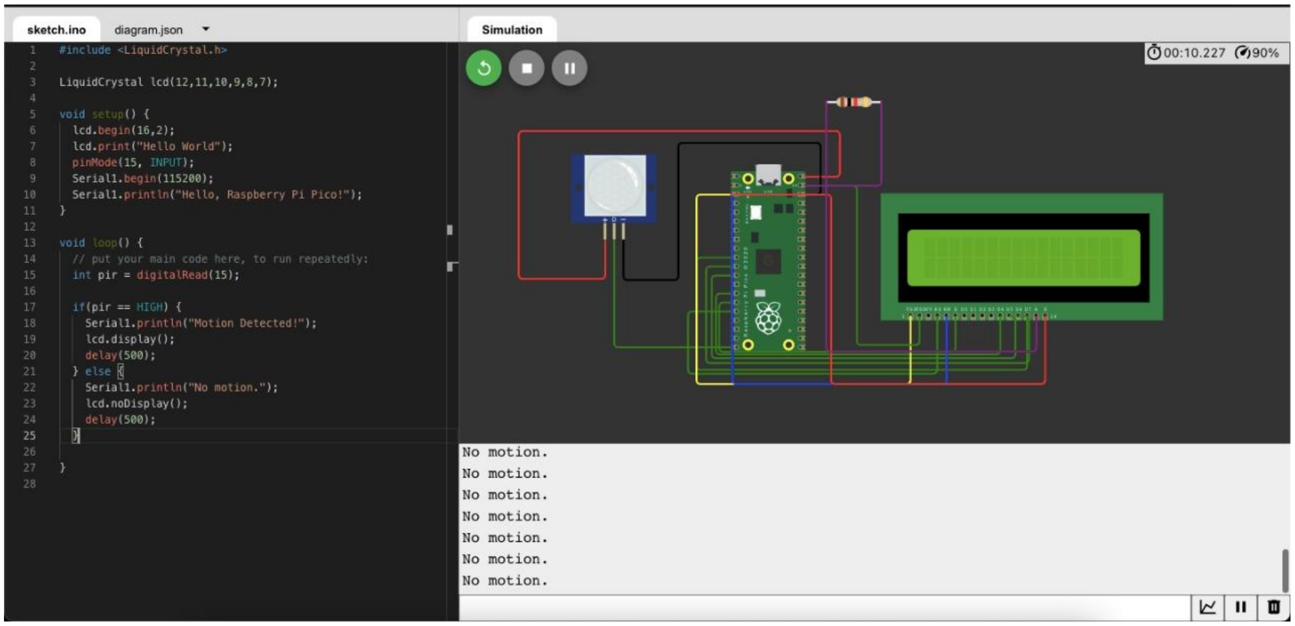
5.2. Software Requirements –

- RASPBIAN OS
- One Weather API
- Google News API
- Python 3.7

5.3. Architecture Diagram

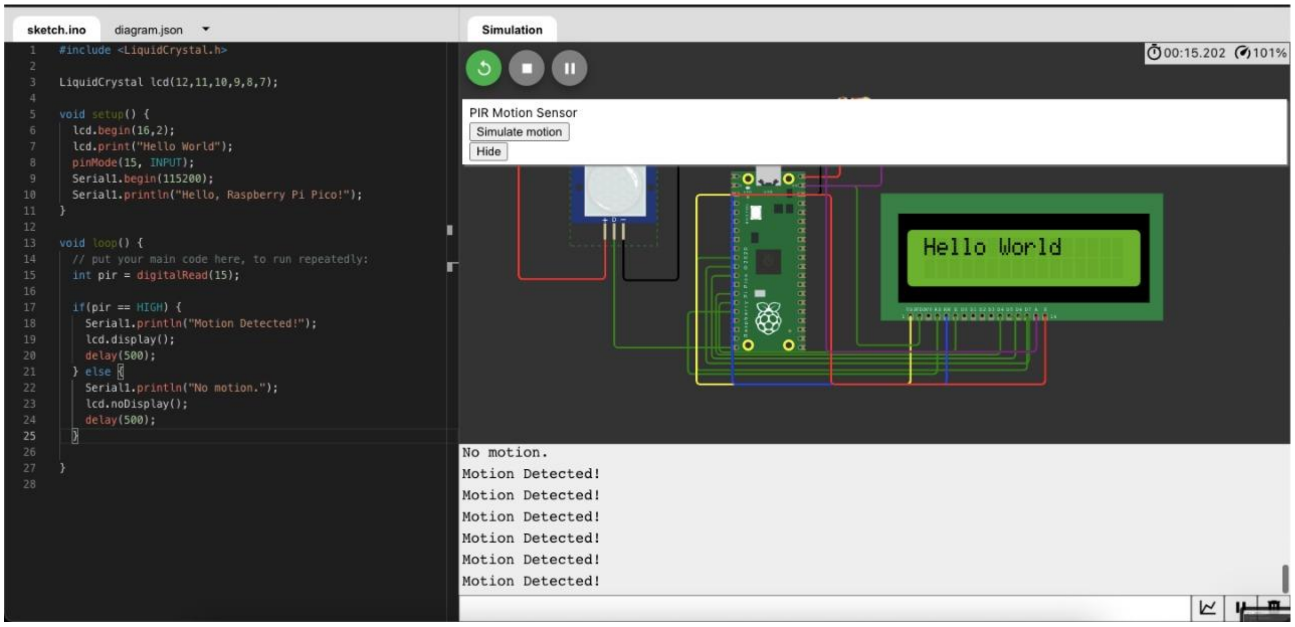


5.4. Circuit Diagrams



```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,10,9,8,7);

void setup() {
  lcd.begin(16,2);
  lcd.print("Hello World");
  pinMode(15, INPUT);
  Serial1.begin(115200);
  Serial1.println("Hello, Raspberry Pi Pico!");
}

void loop() {
  // put your main code here, to run repeatedly:
  int pir = digitalRead(15);

  if(pir == HIGH) {
    Serial1.println("Motion Detected!");
    lcd.display();
    delay(500);
  } else {
    Serial1.println("No motion.");
    lcd.noDisplay();
    delay(500);
  }

}
```

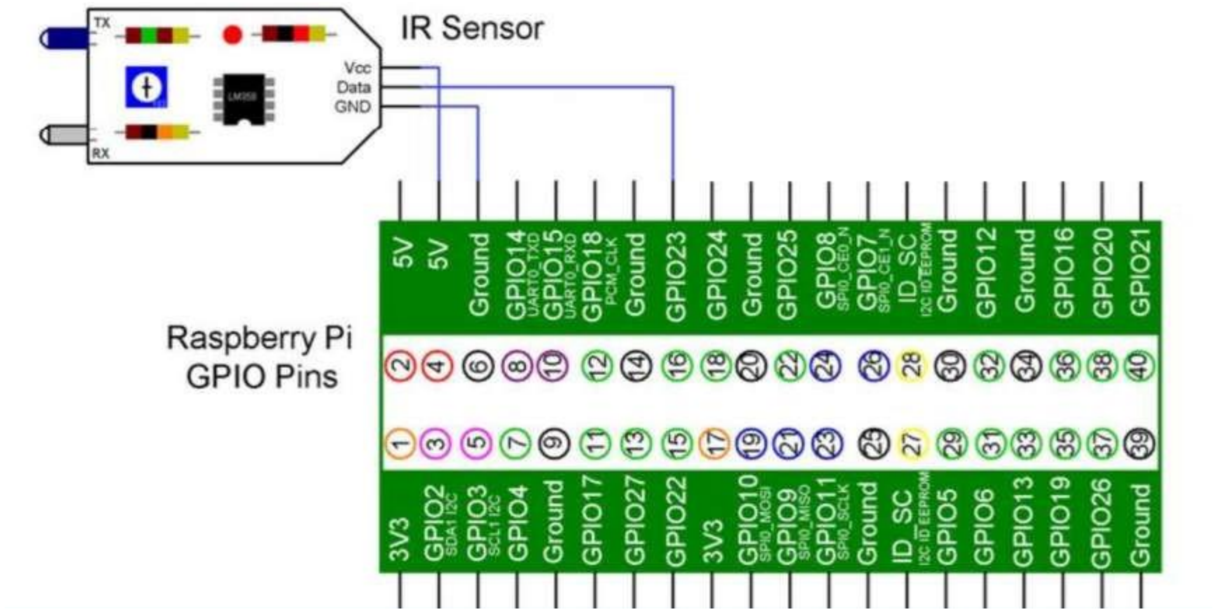
```
No motion.
No motion.
No motion.
No motion.
No motion.
No motion.
No motion.
```

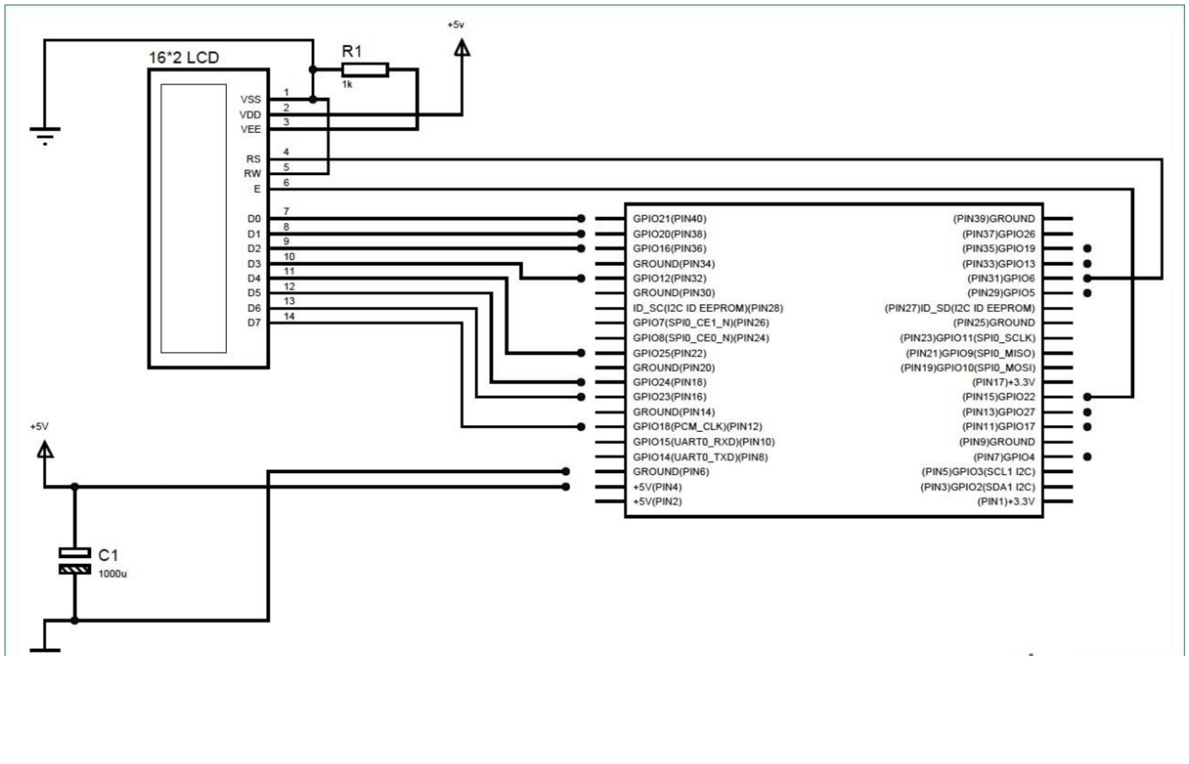


```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,10,9,8,7);

void setup() {
  lcd.begin(16,2);
  lcd.print("Hello World");
  pinMode(15, INPUT);
  Serial1.begin(115200);
  Serial1.println("Hello, Raspberry Pi Pico!");
}

void loop() {
  // put your main code here, to run repeatedly:
  int pir = digitalRead(15);

  if(pir == HIGH) {
    Serial1.println("Motion Detected!");
    lcd.display();
    delay(500);
  } else {
    Serial1.println("No motion.");
    lcd.noDisplay();
    delay(500);
  }

}
```

```
No motion.
Motion Detected!
Motion Detected!
Motion Detected!
Motion Detected!
Motion Detected!
Motion Detected!
```

5.5. Pin Diagrams





6. Implementation

Interfacing PIR sensor with Raspberry Pi

1. Connect the VCC and GND pins of the PIR Motion Sensor to +5V and GND pins of the Rasp-berry Pi.

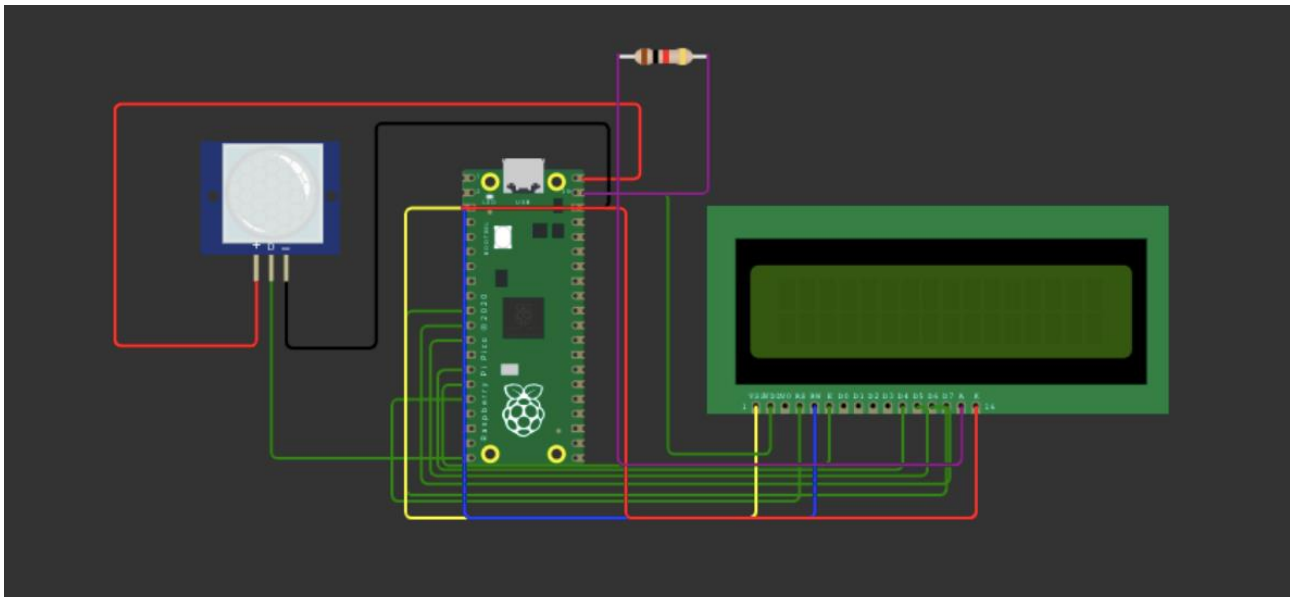
2. Connect the DATA Pin of the PIR Sensor to GPIO23 i.e. Physical Pin 16 of the Raspberry Pi.

Interfacing 16x2 LCD display with Raspberry Pi

1. Connect pins 1 and 16 of the LCD to GND and pins 2 and 15 to 5V supply.

2. The three control pins of the LCD i.e. RS (Pin 4), RW (Pin 5) and E (Pin 6) are connected to GPIO Pin 7 (Physical Pin 26), GND and GPIO Pin 8 (Physical Pin 24).

3. Since we are configuring the LCD in 4 – bit mode, we need only 4 data pins (D4 to D7). D4 of LCD is connected to GPIO25 (Physical Pin 22), D5 to GPIO24 (Physical Pin 18), D6 to GPIO24 (Physical Pin 16) and D7 to GPIO18 (Physical Pin 12).



7. Conclusion

The main goal of the system is to develop a wireless notice board that is easy to install, user friendly system, which can receive and display notice in a particular manner with respect to date and time which will help the user to easily keep the track of notice board every day and each time he uses the system. We have successfully demonstrated the usage of Raspberry PI, IR Sensor and integrated them with the help of Python programming.

8. Appendices

Source Code –

- **pir_motion.ino**
  This file contains the Arduino code for motion detection using a PIR sensor.

```
#include <LiquidCrystal.h>
    LiquidCrystal lcd(12,11,10,9,8,7);

    void setup() {
      lcd.begin(16,2);
      lcd.print("Hello World");
      pinMode(15, INPUT);
      Serial1.begin(115200);
      Serial1.println("Hello, Raspberry Pi Pico!");
    }
    void loop() {
      int pir = digitalRead(15);
      if(pir == HIGH) {
        Serial1.println("Motion Detected!");
        lcd.display();
        delay(500);
      } else {
        Serial1.println("No motion.");
        lcd.noDisplay();
        delay(500);
      }
    }
```

- **smartmirror.py file:** This file contains the Python code for the actual graphical interface of the smart mirror.

```
from tkinter import *
import locale
import threading
import time
import requests
import json
import traceback
import feedparser

from PIL import Image, ImageTk
from contextlib import contextmanager

LOCALE_LOCK = threading.Lock()

ui_locale = ''
time_format = 12 # 12 or 24
date_format = "%b %d, %Y"
news_country_code = 'us'
weather_api_token = '7e60d610eaba669dd9cc2f952e177ebe'
ipstack_key = '25d030228f184b35b2d30aca9751e426'
news_api = '1fe3ec49085c43a89fca27b7cca4c864'
weather_lang = 'en'
weather_unit = 'us'
latitude = None
longitude = None
xlarge_text_size = 94
large_text_size = 48
medium_text_size = 28
small_text_size = 18

@contextmanager
def setlocale(name): #thread proof function to work with locale
    with LOCALE_LOCK:
        saved = locale.setlocale(locale.LC_ALL)
        try:
            yield locale.setlocale(locale.LC_ALL, name)
        finally:
            locale.setlocale(locale.LC_ALL, saved)

icon_lookup = {
    'clear-day': "assets/Sun.png",  # clear sky day
    'wind': "assets/Wind.png",   #wind
    'cloudy': "assets/Cloud.png",  # cloudy day
    'partly-cloudy-day': "assets/PartlySunny.png",  # partly cloudy day
    'rain': "assets/Rain.png",  # rain day
    'snow': "assets/Snow.png",  # snow day
    'snow-thin': "assets/Snow.png",  # sleet day
    'fog': "assets/Haze.png",  # fog day
    'clear-night': "assets/Moon.png",  # clear sky night
    'partly-cloudy-night': "assets/PartlyMoon.png",  # scattered clouds night
    'thunderstorm': "assets/Storm.png",  # thunderstorm
    'tornado': "assests/Tornado.png",    # tornado
    'hail': "assests/Hail.png"  # hail
}


class Clock(Frame):
    def __init__(self, parent, *args, **kwargs):
        Frame.__init__(self, parent, bg='black')
        # initialize time label
        self.time1 = ''
        self.timeLbl = Label(self, font=('Helvetica', large_text_size), fg="white", 
bg="black")
        self.timeLbl.pack(side=TOP, anchor=E)
        # initialize day of week
        self.day_of_week1 = ''
        self.dayOWLbl = Label(self, text=self.day_of_week1, font=('Helvetica', 
small_text_size), fg="white", bg="black")
        self.dayOWLbl.pack(side=TOP, anchor=E)
        # initialize date label
        self.date1 = ''
        self.dateLbl = Label(self, text=self.date1, font=('Helvetica', 
small_text_size), fg="white", bg="black")
        self.dateLbl.pack(side=TOP, anchor=E)
        self.tick()

    def tick(self):
        with setlocale(ui_locale):
            if time_format == 12:
                time2 = time.strftime('%I:%M %p') #hour in 12h format
            else:
                time2 = time.strftime('%H:%M') #hour in 24h format

            day_of_week2 = time.strftime('%A')
            date2 = time.strftime(date_format)
            # if time string has changed, update it
            if time2 != self.time1:
                self.time1 = time2
                self.timeLbl.config(text=time2)
            if day_of_week2 != self.day_of_week1:
                self.day_of_week1 = day_of_week2
                self.dayOWLbl.config(text=day_of_week2)
            if date2 != self.date1:
                self.date1 = date2
                self.dateLbl.config(text=date2)
            # calls itself every 200 milliseconds
            # to update the time display as needed
            # could use >200 ms, but display gets jerky
            self.timeLbl.after(200, self.tick)


class Weather(Frame):
    def __init__(self, parent, *args, **kwargs):
        Frame.__init__(self, parent, bg='black')
        self.temperature = ''
        self.forecast = ''
        self.location = ''
        self.currently = ''
        self.icon = ''
        self.degreeFrm = Frame(self, bg="black")
        self.degreeFrm.pack(side=TOP, anchor=W)
        self.temperatureLbl = Label(self.degreeFrm, font=('Helvetica',
xlarge_text_size), fg="white", bg="black")
        self.temperatureLbl.pack(side=LEFT, anchor=N)
        self.iconLbl = Label(self.degreeFrm, bg="black")
        self.iconLbl.pack(side=LEFT, anchor=N, padx=20)
        self.currentlyLbl = Label(self, font=('Helvetica', medi-um_text_size),
fg="white", bg="black")
        self.currentlyLbl.pack(side=TOP, anchor=W)
        self.forecastLbl = Label(self, font=('Helvetica', small_text_size),
fg="white", bg="black")
        self.forecastLbl.pack(side=TOP, anchor=W)
        self.locationLbl = Label(self, font=('Helvetica', small_text_size),
fg="white", bg="black")
        self.locationLbl.pack(side=TOP, anchor=W)
        self.get_weather()

    def get_ip(self):
        try:
            ip_url = "http://jsonip.com/"
            req = requests.get(ip_url)
            ip_json = json.loads(req.text)
            return ip_json['ip']
        except Exception as e:
            traceback.print_exc()
            return "Error: %s. Cannot get ip." % e

    def get_weather(self):
        try:

            if latitude is None and longitude is None:
                # get location
                location_req_url = "http://api.ipstack.com/%s?access_key=%s&for-
mat=1" % (self.get_ip(), ip-stack_key)
                r = requests.get(location_req_url)
                location_obj = json.loads(r.text)

                lat = location_obj['latitude']
                lon = location_obj['longitude']

                location2 = "%s, %s" % (location_obj['city'], loca-tion_obj['re-
gion_code'])

                # get weather
                weather_req_url = "https://api.openweathermap.org/data/2.5/one-
call?lat=%s&lon=%s&appid=%s&units=metric" % (lat, lon, weather_api_token)
                # weather_req_url = "https://api.darksky.net/fore-
cast/%s/%s,%s?lang=%s&units=%s" % (weath-er_api_token,
lat,lon,weather_lang,weather_unit)
            else:
                location2 = ""
                # get weather
                weather_req_url = "https://api.openweathermap.org/data/2.5/one-
call?lat=%s&lon=%s&appid=%s&units=metric" % (latitude, longitude, weather_api_to-
ken)
                # weather_req_url = "https://api.darksky.net/fore-
cast/%s/%s,%s?lang=%s&units=%s" % (weath-er_api_token, latitude, longitude,
weather_lang, weather_unit)

            r = requests.get(weather_req_url)

            weather_obj = json.loads(r.text)

            # print(weather_obj["hourly"])
            degree_sign= u'\N{DEGREE SIGN}'
            temperature2 = "%s%s" % (str(int(weather_obj['current']['temp'])), de-
gree_sign)
            currently2 = weath-er_obj['current']["weather"][0]["description"]
            forecast2 = weath-er_obj["hourly"][0]["weather"][0]["description"]

            icon_id = weather_obj['current']["weather"][0]["icon"]
            icon2 = None

            if icon_id in icon_lookup:
                icon2 = icon_lookup[icon_id]

            if icon2 is not None:
                if self.icon != icon2:
                    self.icon = icon2
                    image = Image.open(icon2)
                    image = image.resize((100, 100), Image.ANTIALIAS)
                    image = image.convert('RGB')
                    photo = ImageTk.PhotoImage(image)

                    self.iconLbl.config(image=photo)
                    self.iconLbl.image = photo
            else:
                # remove image
                self.iconLbl.config(image='')

            if self.currently != currently2:
                self.currently = currently2
                self.currentlyLbl.config(text=currently2)
            if self.forecast != forecast2:
                self.forecast = forecast2
                self.forecastLbl.config(text=forecast2)
            if self.temperature != temperature2:
                self.temperature = temperature2
                self.temperatureLbl.config(text=temperature2)
            if self.location != location2:
                if location2 == ", ":
                    self.location = "Cannot Pinpoint Location"
                    self.locationLbl.config(text="Cannot Pinpoint Loca-tion")
                else:
                    self.location = location2
                    self.locationLbl.config(text=location2)
        except Exception as e:
            traceback.print_exc()
            print("Error: %s. Cannot get weather." % e)

        self.after(600000, self.get_weather)

    @staticmethod
    def convert_kelvin_to_fahrenheit(kelvin_temp):
        return 1.8 * (kelvin_temp - 273) + 32


class News(Frame):
    def __init__(self, parent, *args, **kwargs):
        Frame.__init__(self, parent, *args, **kwargs)
        self.config(bg='black')
        self.title = 'News' # 'News' is more internationally generic
        self.newsLbl = Label(self, text=self.title, font=('Helvetica', me-
dium_text_size), fg="white", bg="black")
        self.newsLbl.pack(side=TOP, anchor=W)
        self.headlinesContainer = Frame(self, bg="black")
        self.headlinesContainer.pack(side=TOP)
        self.get_headlines()

    def get_headlines(self):
        try:
            # remove all children
            for widget in self.headlinesContainer.winfo_children():
                widget.destroy()
            if news_country_code == None:
                headlines_url = "https://newsapi.org/v2/top-headlines?coun-
try=us&apiKey=%s" % news_api
            else:
                headlines_url = "https://newsapi.org/v2/top-headlines?coun-
try=%s&apiKey=%s" % (news_country_code, news_api)


            r = requests.get(headlines_url)

            feed = json.loads(r.text)
            print(feed)
            for post in feed["articles"][0:5]:
                headline = NewsHeadline(self.headlinesContainer, post["title"])
                headline.pack(side=TOP, anchor=W)
        except Exception as e:
            traceback.print_exc()
            print("Error: %s. Cannot get news." % e)

        self.after(600000, self.get_headlines)


class NewsHeadline(Frame):
    def __init__(self, parent, event_name=""):
        Frame.__init__(self, parent, bg='black')

        image = Image.open("assets/Newspaper.png")
        image = image.resize((25, 25), Image.ANTIALIAS)
        image = image.convert('RGB')
        photo = ImageTk.PhotoImage(image)

        self.iconLbl = Label(self, bg='black', image=photo)
        self.iconLbl.image = photo
        self.iconLbl.pack(side=LEFT, anchor=N)

        self.eventName = event_name
        self.eventNameLbl = Label(self, text=self.eventName, font=('Helvetica',
small_text_size), fg="white", bg="black")
        self.eventNameLbl.pack(side=LEFT, anchor=N)
```

```
class Calendar(Frame):
    def __init__(self, parent, *args, **kwargs):
        Frame.__init__(self, parent, bg='black')
        self.title = 'Calendar Events'
        self.calendarLbl = Label(self, text=self.title, font=('Helvetica', me-
dium_text_size), fg="white", bg="black")
        self.calendarLbl.pack(side=TOP, anchor=E)
        self.calendarEventContainer = Frame(self, bg='black')
        self.calendarEventContainer.pack(side=TOP, anchor=E)
        self.get_events()

    def get_events(self):
        #TODO: implement this method
        # reference https://developers.google.com/google-apps/calendar/quick-
start/python

        # remove all children
        for widget in self.calendarEventContainer.winfo_children():
            widget.destroy()

        calendar_event = CalendarEvent(self.calendarEventContainer)
        calendar_event.pack(side=TOP, anchor=E)
        pass


class CalendarEvent(Frame):
    def __init__(self, parent, event_name="Event 1"):
        Frame.__init__(self, parent, bg='black')
        self.eventName = event_name
        self.eventNameLbl = Label(self, text=self.eventName, font=('Helvetica',
small_text_size), fg="white", bg="black")
        self.eventNameLbl.pack(side=TOP, anchor=E)


class FullscreenWindow:

    def __init__(self):
        self.tk = Tk()
        self.tk.configure(background='black')
        self.topFrame = Frame(self.tk, background = 'black')
        self.bottomFrame = Frame(self.tk, background = 'black')
        self.topFrame.pack(side = TOP, fill=BOTH, expand = YES)
        self.bottomFrame.pack(side = BOTTOM, fill=BOTH, expand = YES)
        self.state = False
        self.tk.bind("<Return>", self.toggle_fullscreen)
        self.tk.bind("<Escape>", self.end_fullscreen)
        # clock
        self.clock = Clock(self.topFrame)
        self.clock.pack(side=RIGHT, anchor=N, padx=100, pady=60)
        # weather
        self.weather = Weather(self.topFrame)
        self.weather.pack(side=LEFT, anchor=N, padx=100, pady=60)
        # news
        self.news = News(self.bottomFrame)
        self.news.pack(side=LEFT, anchor=S, padx=100, pady=60)
        # calender - removing for now
        # self.calender = Calendar(self.bottomFrame)
        # self.calender.pack(side = RIGHT, anchor=S, padx=100, pady=60)

    def toggle_fullscreen(self, event=None):
        self.state = not self.state  # Just toggling the boolean
        self.tk.attributes("-fullscreen", self.state)
        return "break"

    def end_fullscreen(self, event=None):
        self.state = False
        self.tk.attributes("-fullscreen", False)
        return "break"

if __name__ == '__main__':
    w = FullscreenWindow()
    w.tk.mainloop()
```

9. References

1. Belma Ramic-Brkic ,Sadeta Kulovic, (2018), ”Diy Smart Mirror”, Springer International Publishing ,2018.

2. Alexander Nikic Frank Honold,Daniel Besserer, Felix Schüssel , Johannes Bäurle, Michaelweber (2016), “Fitmirror: A Smart Mirror For Positive Affect In Everyday User Morning Routines”, Acm Publication , November 2016.
3. Kamaruzzaman Jahidin , Muhammad Mu’izzudeenYusri , Mohammad Syafwan Ar-shad, Rohayanti Hassan , Shahreen Kasim , Zubaile Abdullah Husni Ruslai (2017),”Smart Mirror For Smart Life“, Ieee Publication,March 2017.

4. Maninder Jeet Kaur, Piyush Maheshwari,SarthakAnand, (May 2017) “Smart Mirror:A Reflective Interface To Maximize Producticity”, International Journal Of Computer Apllications,2017.

5. Adesh Kumar, Anita Gehlot, Mohit Suyal, Nikhil, Rajesh Singh, Vivek Kaundal, (2017), “Microcontroller And Fpga-Based Analysis Of 8 48 Led Matrix Display With Keyboard Interface”,Springer Publication,2017.