IoT Smart Clock

Key Contributors:

Osvaldo Garza, Marcus Harmon, Ruby Rodriguez

Date:

04-21-2022

This project and the preparation of this report were funded in part by ….through an agreement with the University of the Incarnate Word.

Cyber Security Systems and the University of the Incarnate Word

EXECUTIVE SUMMARY

Start of executive summary ……

Our group was able to create a fully functional IoT Smart Clock. Essentially this project aims to tell the user what the current time and date is when the program is runned. Our program defines the code used to display the current time and date in a callable function in which espeak grabs that function, outputs the code by talking then displaying it on the lcd screen for the user to observe.

Project Milestones: E.g. Major steps required to complete your project.

1. Downloading specific packages
2. Enabling LCD screen
3. Programming espeak and lcd to work together

Deliverables: E.g. Report, Deployed architecture, other project outcomes etc.

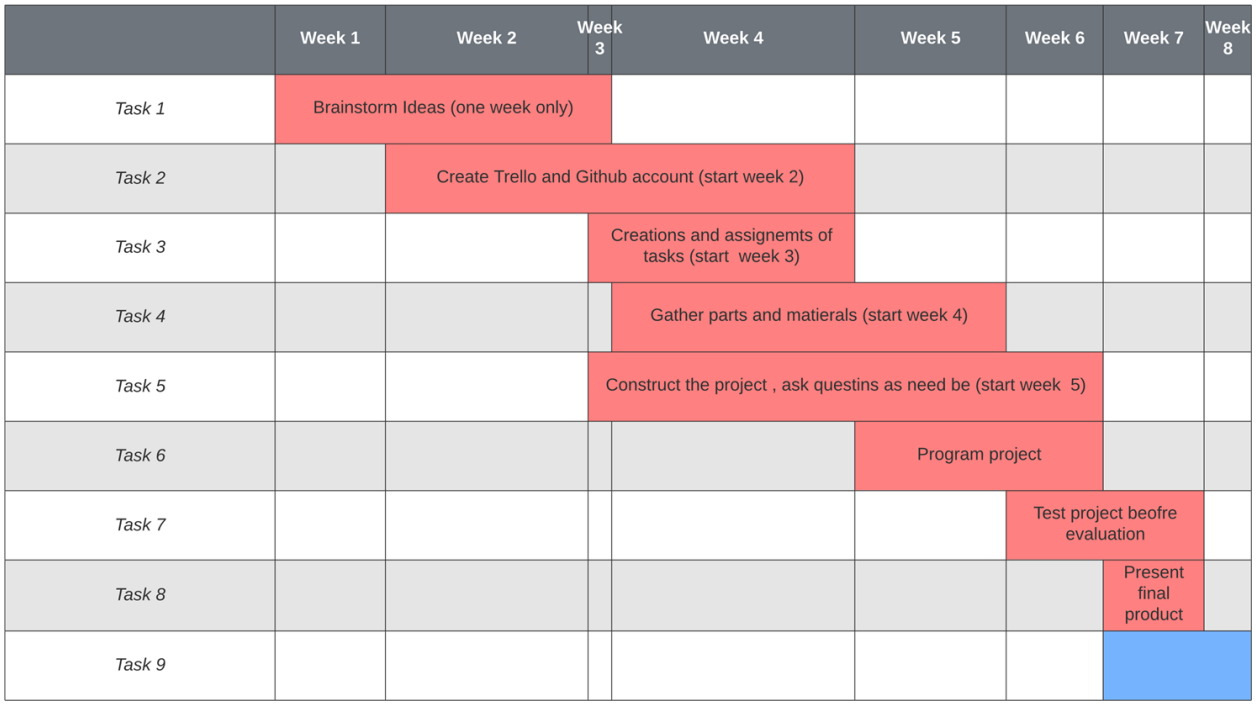
1. Installed Espeak, rpi_lcd, pytz
2. LCD screen fully functional
3. Complete code that fully functions with espeak and LCD to display time and date.

Professional Accomplishments: E.g. New skills that you developed

1. Teambuilding
2. Coding in Pythons
3. Understanding espeak

PROJECT SCHEDULE MANAGEMENT

Create a Gantt chart with the application of your choice and replace it with the picture presented below.



Project Management Board Link (QR Code Only). Send invite to user: @gonzalodparra

Create a Github Project Repository and add the user “cyberknowledge” as a contributor.

marcusharmon4/IoT-Smart-Clock: The date and time is displayed on the LCD screen and it is then said out loud to you with the use of a speaker (github.com)

TABLE OF CONTENTS

Milestones

1. Downloading specific packages

In order to even begin working on the IoT smart clock our group had to download a various amount of packages in order for our code to work. After updating the library we were able to install packages by using the following commands “sudo apt install espeak”, “sudo pip3 install rpi_lcd”, and “sudo pip3 install pytz.” Once we were able to install espeak, rpi_lcd, and pytz we were then able to move on with the rest of our code in order to complete our project.

2. Enabling the LCD screen

One of the key components of the smart clock is the LCD displaying the messages our group wants it to display. However, to even get it to display our team had to do a series of commands in order to activate the LCD. Our team used the command “sudo raspi-config” in order to access the raspberry pi configuration tool and enable the LCD. From there we used the command “sudo reboot” to reboot the entire system. Once the reboot was complete we used the “shutdown now” to prevent system damage to the LCD when finally connecting it to the raspberry pi. Lastly we activate the raspberry pi once again in which the LCD screen and light are now on meaning the LCD screen is now enabled and ready to be programmed.

3. Programming espeak and LCD to work together

Now that our team was able to enable the LCD screen and download the required packages needed for our code to work we now had to test our code in order for espeak and lcd to work together hand in hand. Although, this may be seen as a deliverable instead it’s different to the way we programmed it in order to get our final completed coding result for our project. As a team we were able to construct a code line for time_current in order for it to be said aloud by espeak by having it called time_current and displayed on the LCD screen. Since we were able to create a small program that can be called and spoken by espeak we began to shift our focus on a new line of code that combines both the current time and date to be said by espeak and displayed on the LCD screen. We used this part as a stepping stone in order to fully accomplish our final code for this project.

Deliverables

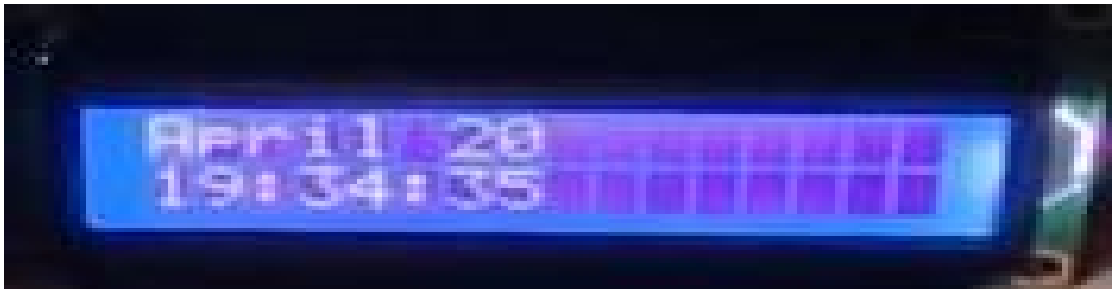
1. Installed Espeak, rpi_lcd, and pytz

Our team successfully installed espeak, rpi_lcd, and pytz which helped fully complete our IoT smart clock code. We were able to use espeak in a correctful manner using a callable function to grab the current time and date needed to be said while rpi_lcd simultaneously displayed the time and date on the lcd screen. However without downloading pytz our completed code itself wouldn't work at all regardless of espeak and rpi_lcd being downloaded because pytz is basically the anchor to our entire code. Pytz is what does the calculations needed for the current timezone and date. Without it our project would be basically useless because pytz tells espeak what it will say aloud and also tells the lcd what it will display.

```
from signal import signal, SIGTERM, SIGHUP, pause
from rpi_lcd import LCD
import time
import datetime
from datetime import datetime as dt2
import subprocess
#from pytz import timezone
import pytz
```

2. LCD screen fully functional

Our team was able to configure and download the rpi_lcd in order to get the LCD screen up and going. After completing our final part of the program to correspond with both the LCD and espeak we then ran the code and successfully were able to have the LCD screen display the current time and date.



3. Complete code that fully functions with espeak and LCD to display time and date

Our first line of code was a stepping stone in order for us to reach the full potential and outcome of our IoT smart clock. Essentially using time_current line code was to test and run with the idea of using a callable function that can be spoken aloud and displayed on the LCD screen which was a complete success. From there our team brainstormed a code to use the pytz package in the callable function in order for it to display through “lcd.text(‘callable_function1’,1) and “lcd.text(‘callable_function2’, 2). Once we were able to create a new complete code that functions with espeak, pytz, and the lcd screen we went ahead and tested it and the output read the current time and date botht through espeak and the lcd screen, an overwhelming success on our part.

Final Complete Code

```
from signal import signal, SIGTERM, SIGHUP, pause

from rpi_lcd import LCD

import time

import datetime

from datetime import datetime as dt2

import subprocess

#from pytz import timezone

import pytz

lcd = LCD()

def execute_unix(inputcommand):

   p = subprocess.Popen(inputcommand, stdout=subprocess.PIPE, shell=True)

   (output, err) = p.communicate()

   return output

def safe_exit(signum, frame):

    exit(1)
```

```
while True:

    t = time.time()

    ### time_current = time.strftime("%H:%M:%S", time.gmtime(t))

    my_datetime = dt2.now(datetime.timezone.utc)

    ### my_datetime_cst =
my_datetime.astimezone(pytz.timezone('US/Central')).strftime('%Y-%m-%d %H:%M:%S
%Z%z')

    my_datetime_cst_1 = my_datetime.astimezone(pytz.timezone('US/Central')).strftime('%B
%d')

    my_datetime_cst_2 =
my_datetime.astimezone(pytz.timezone('US/Central')).strftime('%H:%M:%S')

    print(my_datetime)

    print(my_datetime_cst_1)

    print(my_datetime_cst_2)

    try:

        # Display date and time in LCD

        lcd.text(my_datetime_cst_1, 1)

        lcd.text(my_datetime_cst_2, 2)

        # Create Phonic Messages

        msg_1 = 'espeak -ven+m3 -k5 -s150 --punct="<characters>" "%s" 2>>/dev/null' %
my_datetime_cst_1 #speak aloud

        msg_2 = 'espeak -ven+m3 -k5 -s150 --punct="<characters>" "%s" 2>>/dev/null' %
my_datetime_cst_2 #speak aloud

        # Execute Phonetic Messages

        execute_unix(msg_1)

        execute_unix(msg_2)

        signal(SIGTERM, safe_exit)
```

```
    signal(SIGHUP, safe_exit)
    #time.sleep()
    pause()
except KeyboardInterrupt:
    pass
finally:
    lcd.clear()
```

Accomplishments

1. Teambuilding

Our team gained a better understanding of working together as one. Working as a team allowed us to explore different ideas and solutions to tackle problems or overcome challenges we faced together while working on this project. Working as a team also helped strengthen our communication skills, problem solving skills, time management skills, collaboration skills, and critical thinking skills. The skills we listed above are what really pushed us beyond our boundaries and comfort zones in order to learn new aspects when it comes to working with a raspberry pi.

2. Coding in Python

Before taking this class our team has had prior experience coding in python. However, this project helped test our ability and knowledge of pythons as our project seems to require or know an intermediate level experience of pythons since our complete code does involve multiple packages being used in multiple lines of code we used to complete this project. Overall we were able to understand and basically adapt to this new field of pythons that we have not been shown before and it was interesting having to research certain codes we have never seen before just for those codes to fit or connect with our very own lines of code to receive the desired output we needed for the lcd and espeak. This project definitely helps expose us to a new reality when it comes to coding in which it’s okay to feel uncomfortable, frustrated, and perplexed because it’s a part of the learning process.

3. Understanding espeak

When first learning about our IoT project we were exploring the idea of using espeak for our IoT smart clock, but we really had no knowledge of what espeak is or how to install it until we began reading about it on github as well as finding videos about it on YouTube. Our team learned that espeak is a software speech synthesizer that helps linux communicate with the raspberry pi or it can be used in the linux terminal. Our team was fascinated to hear espeak talk through the raspberry pi when we first ran a command to test if it would work on the linux terminal by simply stating “espeak “Hello Ozzy, Marcus, and Ruby!” We continued to work with espeak by researching new ways to call it through the thonny python program in which we were successfully able to do considering we were able to import pytz and include it in a callable function for espeak to state the timezone and datetime within the code pytz was located in. Overall we were glad to share this new experience with espeak pythons and continue to work it if given the opportunity to do so. Our team has recently discovered that the user can change the language espeak talks to in almost any language in the world, change the output pitch of the voice, and even change the gendered voice of espeak from male to female and vice versa.

Resources

- http://robsraspberrypi.blogspot.com/2016/02/raspberry-pi-all-things-date-and-time.html
- https://www.youtube.com/watch?v=5uBqkFuYaCU&t=294s
- https://howchoo.com/pi/how-to-make-a-raspberry-pi-smart-alarm-clock
- https://www.youtube.com/watch?v=5uBqkFuYaCU
- https://github.com/mheidenreich/LCDDemo/blob/main/lcd-hello.py
- https://www.dexterindustries.com/howto/make-your-respberry-pi-speak/
- https://stackoverflow.com/questions/17547531/how-to-use-espeak-with-python
- https://unix.stackexchange.com/questions/506412/how-can-i-read-out-a-text-file