I. Introduction

(I) Research Motivation

During the pandemic, while attending online school for two years, programming was introduced as a way to fill spare time. Started with HTML and CSS, then expanded to many programming languages such as Python, JavaScript, and PHP. Alongside programming, various software and computer components were explored as a hobby. During the pandemic, a lot of changes for Muslims to practice their faith, many use Zoom and WhatsApp for prayers and live recitals from Masjid. One solution that particularly intrigued me was using Mumble as an audio server, which allowed many people to connect via an Android app and listen to recitations. inspires me in how technology could support community connections.

After moving to Taiwan, where Muslims are a minority, the continued value of this practice became clear. The idea of a portable Adzan caller, live streaming prayers from a Masjid to subscribers, emerged as a way to help Muslims in non-majority Muslim countries maintain their connection to Masjid as the house of prayer. This innovation could bridge the gap for those with limited access to a Masjid, ensuring that they remain connected to their spiritual practices.

The portable Adzan caller is a way for Muslims who live in a country like Taiwan, where Muslims are a non-majority, to stay connected to their faith without breaking local laws. In many Muslim-majority countries, the Adzan "أذان"(call to prayer) is usually announced through loudspeakers. However, in Taiwan, there are strict rules against loud noises (Chou et al., 2012), and breaking these rules can result in fines ranging from NT $3,000 to NT$30,000 (Ministry of Justice, 2010). The portable Adzan caller solves this by streaming the Adzan directly to people's phones or portable devices so they can hear it without disturbing others.

The portable Adzan caller is more convenient than other options like YouTube, Google Nest, and Amazon Alexa. YouTube causes stream delays and is not created for live religious events, and Google Nest and Alexa are more for personal use in addition to getting a straw as a high-quality audio stream to a very large audience.

Moreover, in Islam, the Adzan is considered to be much more valuable when it's done by a person and not played from a recording as this keeps the spiritual bonding solid. Therefore, the portable Adzan caller will help adzan that was again aired from a Masjid, and it followed the tradition to help Muslims develop their faith in areas where loudspeakers are not allowed.

(II) Research Purpose

A. Exploring Local Noise Regulations

Research how local noise rules in non-Muslim majority nations affect the freedom of Muslims to carry out the religion openly, as it relates to the Adzan. Discuss the social, legal, and cultural implications of such restriction and identify other ways that the religious and community significance of the Adzan can be ensured without violating local laws. This study will seek to give an understanding of how such issues can be met through technology solutions without compromising religious practice or causing offense in the wider community.

B. Develop the technical architecture of the portable Adzan caller.

Explore how to use Raspberry Pi, the Barnard Mumble server, and the tmux system to implement a portable Adzan caller that can stream real-time audio in non-Muslim majority countries. Design and implement a system that enables real-time Adzan broadcasting while respecting local. This includes the explanation of the hardware, software, and network infrastructure required to create a user-friendly, low-latency, and scalable solution that can be adopted by Muslim communities in diverse environments.

II. Literature Review

(I) The Adzan: Historical and Cultural Context

The Adhan, or the call to prayer, is profoundly significant to the Muslim community on a day to day basis. As cited by Sahih al-Bukhari (870). when the Muslims first reached Medina, prayer times had no formal announcement. It was not until the use of bells, that the Prophet Muhammad (peace be upon him) was able to accepted a human voice suggestion by Umar. Bilal ibn Rabah was appointed to make the first Adhan, which since has turned out to be one of the basic features of worship in Islam (Al-Bukhari, 870)

Beyond its ritualistic function, the Adzan serves as a spiritual and communal bond among Muslims. But, in non-Muslim-majority countries, broadcasting the Adzan through loudspeakers is often restricted due to noise regulations. Historically, restrictions on public Muslim religious expressions, including the Adzan, date back to the Council of Vienna (1311-1312) (Constable, 2010). Today, similar problems exist, most importantly in urban areas where traffic, construction, and nightlife already contribute to noise pollution. While religious freedom is protected in many places, the Adzan's public broadcast can be controversial, particularly in communities unfamiliar with the practice.

Several ways have been created, including Islamic mobile applications (e.g., Muslim Pro) and radio-based Adzan broadcasting. Although, these methods lack the live human component, which is spiritually significant in Islam. In response, the Portable Adzan Caller, as its name indicates, tries to reproduce this ritual through modern means by booming the Adzan at places which traditional norm forbids, traditionally loud speaker to mention in few countries. This project seeks to appreciate the Adzan in its historical context while at the same time preserving its spiritual values through the use of technology.

(II) The Role of Technology in Religious Adaptation

It is not new to adopt a technology in religious practices. Virtual churches and e-worship help worshipers who are unable to attend worship physically (Sadiku et al., 2022), Jewish Sabbath Clocks where technology helps provide reminders for prayer without violating Sabbath rules (Bix, 2020), and Digital Pilgrimage guidebooks for Muslim pilgrims through augmented reality and mobile applications in navigating Hajj ceremonies (Binsawad et al., 2022). These examples reveal the central role of technology in maintaining religious tradition while being within the bounds of modern legal regulations.

The Portable Adzan Caller does the same by allowing Muslims residing in minority countries to remain connected to their faith without violating local noise regulations.

Apps like Muslim Pro, Athan, and iPray provide Adzan alerts but through recorded voice instead of a live Muadhin, and that makes it less spiritual and less connected to the actual prayer time of the Masjid. Since each person is getting notifications based on his/her own phone settings, there isn't the sense of praying together.

Smart assistants like Google Nest and Alexa can also play Adzan, but they only work for individuals at home, not for a whole community. FM radio has been used in some places to broadcast the Adzan, but not everyone wants to hear it, making it unsuitable for public use. A better approach would be a subscriber-based system where only people who want to hear the Adzan can receive it privately through digital radio or mobile-connected receivers.

All of these solutions prove the complexity of having a live Adzan caller in Muslim-minority nations. An ideal system should play Adzan live and at the same time for all individuals irrespective of where they are such as home, workplace, or outside. Most importantly, it should allow Muslims to feel united through hearing Adzan at the same time while being respectful to the people around them who may not want to hear it.

(III) IoT-Based Religious Solutions

Recently, the Internet of Things (IoT) technology has established a real-time Adzan broadcasting. One example is the IoT-based Adzan system (Ahyar et al.,2024) uses a live streaming system that connects the mosques to user devices via IceCast, a digital radio broadcasting technology. However, their system requires touchscreen interaction to manage the broadcast, which is not ideal for mosque administrators who need a more simple solution.

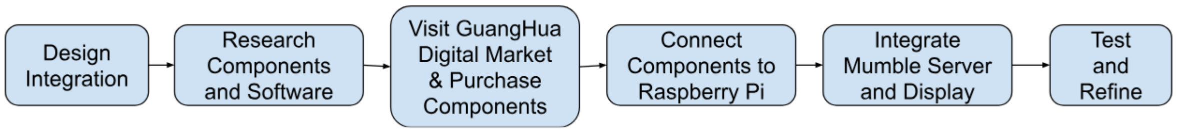
Using a different approach, the Portable Adzan Caller uses Mumble, a low-latency voice communication platform originally made for a real-time gaming audio. Unlike IceCast, which operates as a radio-based streaming service with high chance of buffering delays, Mumble is optimized for instant voice transmission with minimal lag. Instead of requiring a touchscreen interface, this system operates with a simple on/off switch, making it more intuitive for the Muadhin to start and stop the Adzan. Additionally, Mumble supports encrypted, subscriber-based access, ensuring that only those who choose to receive the Adzan will hear it which is the key advantage in non-Muslim-majority environments where public broadcasting may not be feasible.

By integrating real-time streaming with a user-friendly design, the Portable Adzan Caller offers a practical solution for Adzan practice in countries where loudspeakers are forbidden. Such systems keep the Adzan live using IoT to bridge Muslims with their religious practice while remaining compliant with local laws and cultural sensitivities.

III. Research Method

(I) Raspberry Pi Design

Figure 1: Process of Integrating Raspberry Pi Components


Design Integration
Research Components and Software
Visit GuangHua Digital Market & Purchase Components
Connect Components to Raspberry Pi
Integrate Mumble Server and Display
Test and Refine

Source of Figure 1: Self-made

As shown in Figure 1, the design of Portable Adzan Caller was step by step. The integration of the client (Adzan listener) and the caller (Muadhin) began to ensure the audio streaming was smooth. The communication between the Adzan Caller, the Raspberry Pi, the Mumble server, and the users' devices was created to ensure the data transmission was smooth. Experiments were conducted with the aid of online forums and source code hosting websites such as GitHub to identify the required hardware and software. A visit was also taken to Guanghua Digital Market to observe digital shops and buy the required components.

After the necessary parts were gathered, the assembly began by mounting them on the Raspberry Pi. This included setting up the audio output, mounting the I2C display to show the connection status, and linking the button for the "on-air" and "off-air" modes.

After the hardware installation was complete, the software installation began. This involved installing the Mumble server, installing and downloading the client application on the Raspberry Pi, and developing special system control management scripts. The final one was testing and ensuring all parts work in harmony with each other and indeed transmit the Adzan properly with the correct presentation on the I2C display, and trigger the button to run as intended. Through this careful process, a functional and practical Portable Adzan Caller was achieved.

(II) Implementation

A. Hardware Installation

The Table 1 provided outlines the pin connections between the Raspberry Pi 4 B, an OLED display, and a button. For the button, it is connected to GPIO 22 on the Raspberry Pi, ensuring that when pressed, the button links the OUT (Signal) pin to the 3.3V (VCC) pin, which will be read as '1' in the Python program. This setup allows for straightforward logic level detection. It's critical to note that the OLED is sensitive to voltage, and using the wrong voltage could cause damage.

To avoid this, I initially used a breadboard, which offered flexibility and minimized the risk of damaging the OLED during the prototyping phase.

Table 1: Pin Mapping for Raspberry Pi 4 B, OLED, and Button

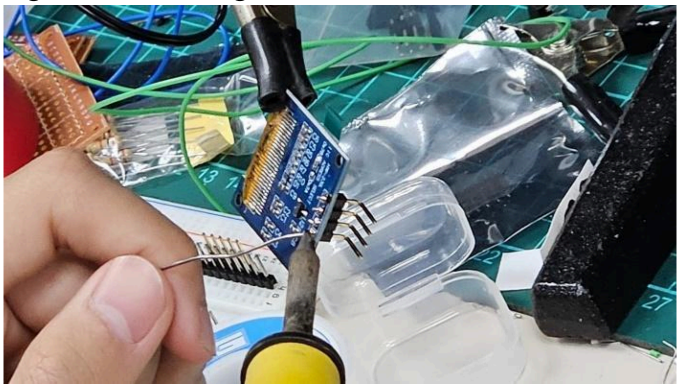
Raspberry Pi	**Pin Number**	**OLED Pins**	**Button Pins**
GND	Pin 6 or Pin 9	GND	GND
3.3V or 5V	Pin 1 (3.3V) or Pin 2 (5V)	VCC	VCC
GPIO 3 (SCL)	Pin 5	SCL	-
GPIO 2 (SDA)	Pin 3	SDA	-
GPIO 22	Pin 15	-	OUT (Signal)

Source of Table 1: Self-made

Once all the Raspberry Pi pins were assembled using a breadboard and the connections were verified, jumper wires were introduced to simplify the setup and reduce complexity.

Then for audio input, I continue using a USB microphone which is an off-the-shelf product that plugs into the raspberry pi via USB and does not require any wiring to the GPIO pins. Hence, this configuration should be straightforward to maintain and to be employed by the adzan caller in the entire project.

Figure 2: Soldering the LED



Source of figure 2: Self-made

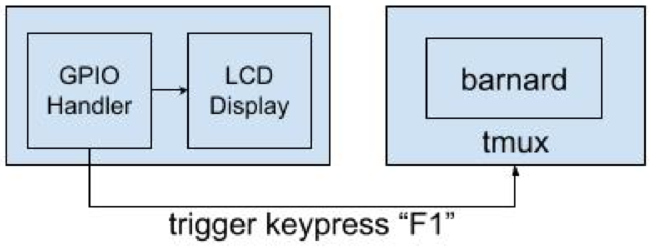
Figure 3: Connecting the button

Source of figure 3: Self-made

B. Software Installation & Programming

This system consists of 2 big parts, as we can see in Figure 4. Part 1 is the button and OLED status display to be the controlling mechanism. And Part 2 is the barnard client as audio publisher which runs inside tmux virtual terminal. Basically, instruction from Part 1 triggers the key F1 inside the tmux virtual terminal, so the barnard program would mute-unmute the microphone.

Figure 4: Software Integration between GPIO, LCD, and Audio Publisher



Source of Figure 4: Self-made

To compile Barnard from source code, a Golang compiler is required. Since Golang is not installed by default on the Raspberry Pi, it must be compiled from the repository, which can be downloaded from the official website.

The website provides instructions on how to download the appropriate version for the system and install it. According to the compilation article (Morgan, 2020), installing Golang on a Raspberry Pi requires downloading the specific Go binary for the ARM architecture, extracting the downloaded files, and configuring the environment variables to allow the system to locate the Go compiler and its components.

C. Off-Air/On-Air Function Using GPIO

Use Python's GPIO library to detect button presses and toggle the OLED display between "OFF AIR" and "ON AIR". Example code snippet for GPIO setup:

Figure 5: Python GPIO Button Script

```python
import RPi.GPIO as GPIO
from subprocess import call
import time

# GPIO setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(22, GPIO.IN, pull_up_down=GPIO.PUD_UP)
toggle_state = False

def send_f1_keypress():
    global toggle_state
    call(["tmux", "send-keys", "-t", "barnard_session", "F1"])
    toggle_state = not toggle_state
    return "OFFLINE" if toggle_state else "ONLINE"

def button_handler(update_text_callback):
    while True:
        if GPIO.input(22) == GPIO.HIGH:
            text = send_f1_keypress()
            update_text_callback(text)
            time.sleep(0.2)
```

Source of figure 5: self-made

D. OLED as Information Screen

Use the Adafruit SSD1306 library to program the OLED screen. Display dynamic information such as "ON AIR" or "OFF AIR" based on button state. Example code for displaying text:

Figure 6: Python GPIO Button Script

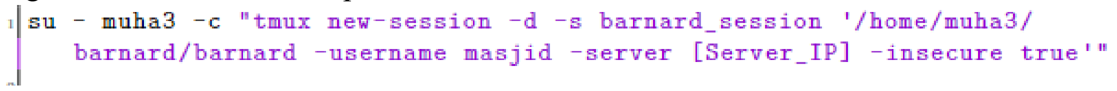
```python
import board, busio, digitalio
from PIL import Image, ImageDraw, ImageFont
import adafruit_ssd1306

# OLED setup
oled_reset = digitalio.DigitalInOut(board.D4)
i2c = board.I2C()
oled = adafruit_ssd1306.SSD1306_I2C(128, 64, i2c, addr=0x3C, reset=
   oled_reset)
oled.fill(0); oled.show()
image = Image.new("1", (oled.width, oled.height))
draw = ImageDraw.Draw(image)
font = ImageFont.truetype('PixelOperator.ttf', 16)
line_text = [""] * 3

def update_display():
    draw.rectangle((0, 0, oled.width, oled.height), outline=0, fill=0)
    for i, text in enumerate(line_text):
        draw.text((0, i * 16), text, font=font, fill=255)
    oled.image(image); oled.show()

def update_text(text):
    line_text[0] = text
    update_display()

if __name__ == '__main__':
    threading.Thread(target=button_handler, args=(update_text,), daemon=
   True).start()
    while True:
        time.sleep(1)
```

Source of figure 6: self-made

E. Linux Scripting for Auto Start and Integration

Start a new tmux session for the Mumble client using tmux: Create a script to start multiple applications (e.g., Barnard, OLED display update script) as background processes. Example tmux session script.

Figure 7: tmux session script



```
su - muha3 -c "tmux new-session -d -s barnard_session '/home/muha3/
   barnard/barnard -username masjid -server [Server_IP] -insecure true'"
```

Source of Figure 7: Self-made

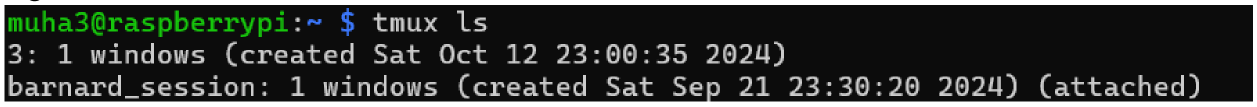
IV. Analysis and Results

(I) Testing the Portable Adzan Caller

To make sure that the Portable Adzan Caller works 24/7, I did an experiment to maintain the device connection to keep connected to the mumble server for 7 days straight without dying. My goal was to evaluate whether the system could maintain the connection to the server without any interruption.

During my 7-day test, the device remained connected and still alive without crashed of overheating or any fatal system as you can see from Figure 8 below that the barnard_session which is the connection from the Portable Adzan Caller to the Mumble server, has been active since September 21 and was still running when checked on October 12. This showed that the system successfully maintained the connection and the durability.

Figure 8: tmux list of sessions



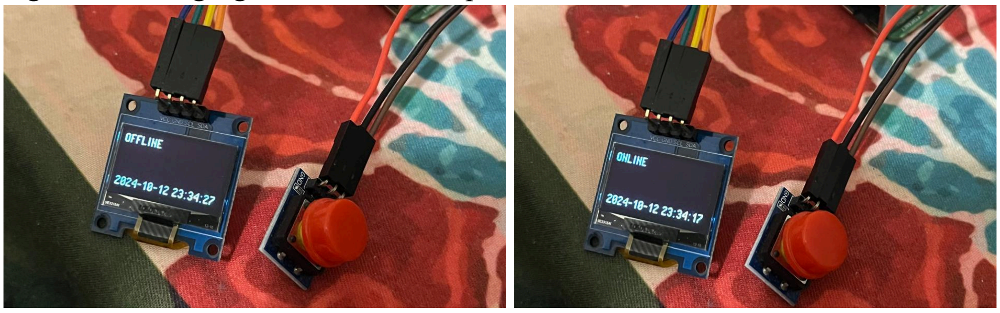
```
muha3@raspberrypi:~ $ tmux ls
3: 1 windows (created Sat Oct 12 23:00:35 2024)
barnard_session: 1 windows (created Sat Sep 21 23:30:20 2024) (attached)
```

Source of Figure 8: Self-made

(II) Testing the LED and Audio Functionality

In addition to testing the connection to the Mumble server, testing was also conducted on the LED display and audio stream quality to ensure that the Portable Adzan Caller functioned as intended. The LED, connected via GPIO, was designed to indicate the system's status, switching between "ON AIR" and "OFF AIR" based on button presses. The button was tested multiple times to verify its functionality, and the results were consistent and reliable, as shown in Figure 9.

Figure 9: Changing mode on button press



Source of Figure 9: Self-made

The final feature that was tested was the audio functionality to verify that the device could stream the Adzan in a clear and less delayed format. The test resulted in positive feedback, with the average latency being 218 ms, indicating that the audio streaming has a delay of approximately 218 milliseconds per packet round-trip. Compared to the IoT-based Adzan system developed by (Ahyar et al., 2024) that adopts a live streaming system with audio delays of up to an average of 3.4 seconds, this system showed significantly less latency, constituting a more responsive solution for real-time Adzan broadcasting. This easy solution offers simple access, with the administrators of mosques able to operate the device as simply as switching a microphone on and off.

V. Conclusion and Suggestions

(I) Conclusion

The Portable Adzan Caller is a way for Muslims that lives in a non major muslim country which can be streamed by real time voice from the Masjid, using Raspberry Pi and the Barnard Mumble server. Using tmux as the system to allow manage multiple sessions, and the OLED screen as the mechanism to change mode on “ON AIR” and “OFF AIR” on button press. The result is quite good, but there is still a lot of area to improve it such as the delay. Throughout this project, I learned a lot of things, from improving my arduino’s component knowledge to testing the device.

(II) Suggestions

In the future, I want to improve the device’s connection to the mumble server, and also I want plan to implement an Artificial Intelligence to the device so it can control the device in a comfortable way, along with the installation of voice command using Artificial Intelligence to provide a more intuitive and efficient control system. For example, since the Mumble server has grouping capabilities, AI could be used to manage subscribers by moving them to different groups. This would make operating the device more intuitive and user-friendly.

VI. References

Al-Bukhari, M. I. (870). *Sahih al-Bukhari: The book of call to prayer (Adzan)* (Vol. 1, Book 11, Hadith 578). Darussalam.

Constable, O. R. (2010). *Regulating religious noise: The Council of Vienne, the mosque call, and Muslim pilgrimage in the late medieval Mediterranean world. Past & Present, 208*(1), 3-32. https://doi.org/10.1163/138078510X12535199002677

Chou, L., Yu, C., & Chen, C. (2012, April 1). *Noise regulations and control policy in Taiwan. Journal of the Acoustical Society of America, 131*(4_Suppl), 3505. https://doi.org/10.1121/1.4709245

Morgan, J. (2020, June 2). *How to install Golang on a Raspberry Pi*. Jeremy Morgan Tutorials. Retrieved August 2024, from https://www.jeremymorgan.com/tutorials/raspberry-pi/install-go-raspberry-pi/

Ministry of Justice, Republic of China (Taiwan). (2010, March 11). *Noise control act* (Article 31). *Laws & Regulations Database of the Republic of China*. Retrieved August 2024, from https://law.moj.gov.tw/ENG/LawClass/LawAll.aspx?pcode=O0030002

Sadiku, M. N. O., Chukwu, U. C., Ajayi-Majebi, A., & Musa, S. M. (2022). *Digital church: An introduction*. *International Journal of Trend in Scientific Research and Development, 6*(4), 695-701. https://www.ijtsrd.com/papers/ijtsrd52538.pdf

Bix, A. S. (2020). *‘Remember the Sabbath’: A history of technological decisions and innovation in Orthodox Jewish communities*. *History and Technology, 36*(2), 205–239. https://doi.org/10.1080/07341512.2020.1816339

Binsawad, M., & Albahar, M. (2022). *A technology survey on IoT applications serving Umrah and Hajj*. *Applied Computational Intelligence and Soft Computing, 2022*, 1919152. https://doi.org/10.1155/2022/1919152

Ahyar, M., Parenreng, M. M., Raharjo, M. F., Nirwana, H., & Abduh, I. (2024, July 10). *Developing call-adzan: A live streaming system for broadcasting the prayer calls*. *AIP Conference Proceedings, 3140*(1), 040019. https://doi.org/10.1063/5.0221052