Remote controlled box

Introduction

This project revolves around a box that can be opened remotely over a Wi-Fi connection. The box hosts a server locally and waits for the client to press a button on the webpage in order to transition from the CLOSED to the OPEN state, or OPEN to CLOSED if it has transitioned previously. Everything will be hosted on the same network, a mobile hotspot for the sake of the project.

General description

Initially, the box is in the OPEN state, where the servo is at a 90 degree angle, the display has the “OPENED” text, IP address and the box is considered opened(lid up). After the user presses the button on the page, the box plays a sound, the motor activates to push down the cover of the box and the displayed message changes to “CLOSED”. A toggle will revert to the initial state.

Hardware Design

Hardware components

- **ESP32 Devkit with ESP-WROOM-32 XX5R69**
- **Servo motor SG90**
- **Active buzzer 3V**
- **OLED 0.96” I2C Display**
- **USB cable (for developing)**
- **Breadboards**
- **Dupont cables**
- **Accumulator 3,7V - 5000 mAh**
- **Cardboard**
- **Sticks/something to fuse the servo with the lid**
- **Glue tape**

Bill of materials

Nr.	Component	Link	Datasheet
1	18650 Accumulator 3.7V 5000mAh	https://www.emag.ro/acumulator-li-ion-3-7v-5000-mah-tip-18650-cu-fire-acum-fire/pd/DW190DMBM/	
2	ESP32 Devkit with ESP-WROOM-32 (XX5R69)	https://www.emag.ro/placa-esp32-cu-esp-wroom-32-30-pini-usb-tip-c-3874784221589/pd/D0JH59YBM/	https://lastminuteengineers.com/esp32-pinout-reference/
3	Servomotor SG90	https://www.optimusdigital.ro/ro/motoare-servomotoare/26-micro-servomotor-sg90.html	https://www.friendlywire.com/projects/ne555-servo-safe/SG90-datasheet.pdf
4	Display OLED 0.96" I2C	https://www.emag.ro/ecran-oled-0-96-ai409-s322-323-324/pd/D69S02MBM/	https://www.mouser.com/datasheet/2/1398/Soldered_333099-3395096.pdf
5	Active buzzer 3V	https://www.optimusdigital.ro/ro/audio-buzzere/635-buzzer-activ-de-3-v.html	
6	400 pins breadboards	https://www.optimusdigital.ro/ro/prototipare-breadboard-uri/44-breadboard-400-points.html	
7	Dupont cables 10-20cm	"Borrowed" from friends ;)	
8	Charger for 18650 batteries	https://www.emag.ro/incarcator-pentru-2-acumulatori-18650-li-ion-conectare-la-priza-220v-negru-32010609/pd/DHH2MWMBM/	
9	USB-C cable	https://www.optimusdigital.ro/ro/cabluri-cabluri-usb/2652-cablu-usb-31-tip-c-catre-usb-20-am.html	

Hardware functionalities

- **Microcontroller: WiFi, PWM, I2C, serial communication, low power (about 100 mA) and cheap, with breadboard ready pins.**
- **Servomotor: Used with PWM at pin GPIO23 to lift the lid. Uses at most 500 mA but only when acted upon.**
- **Display: Shows the IP of the server and also the state of the box. SDA is connected to GPIO21 and SCL TO GPIO22. Low power, uses less than 50 mA.**
- **Active buzzer: Makes sound when the box is transitioning to the open state and back, connected to GPIO14 (any pin that can be set from low to high would do). Uses around 50mA.**
- **Battery: Supplies the whole circuit. Has enough power so that the servo can function multiple times. It's also very efficient in the idle state, as only the display and microcontroller are active most of the time.**

Accumulator, Servo, Display, Buzzer and Microcontroller.

Software Design

Status

The current implementation includes all the functionalities:

- Wi-Fi connection setup (using ssid + password, requires upload to change them)
- Hosting a web server on port 80
- Control of a servo motor to lift up and down the lid of the box
- Displaying IP and lock status on a display
- Buzzing whenever the state of the box changes
- Web interaction with the ESP32

Libraries

Libraries used are:

- WiFi.h – Library for WIFI connection
- Wire.h – Library used for I2C communaction for display
- Adafruit_GFX.h – Library for drawing functions used by the display
- Adafruit_SSD1306.h – OLED driver library for SSD1306-based displays
- ESP32Servo.h – Library used by the servo to make use of PWM

What sets the project apart

It is a very straight forward project, making use of multiple libraries and concepts (WiFi communication, HTML design, PWM, I2C, GPIO, UART). For end users, it's very simple to use, as the IP is clearly displayed, and could be even easier to connect to using a DNS. There's no need for them to do anything other than charge the battery every now and then as long as the network stays the same.

Labs used

- GPIO - For buzzer
- UART - Testing and for making sure board is actually connecting + displaying the correct IP
- PWM - To move the servo a number of degrees
- I2C - To allow communication with the display
- Timers and interrupts - To treat client timeouts

Calibration

The servo was calibrated according to the first link in the software section and does a 90 degree turn. The display was calibrated using the 3rd link. The timeout is 10s, because a mobile hotspot could be a bit slow at times, in a real scenario this should be <1s.

Functions

- onTimeout() - interrupt function that sets the timeout flag to true
- startClientTimer() - function that resets and enables timer

- buzz(int duration) - makes a sound for the provided duration
- moveServo() - alternates between 0 and 90 degrees depending on the state of the box
- updateDisplay() - draws the IP and current state of the box
- toggleLock() - function that applies the previous changes
- setup() - sets up all components and the server
- loop() - handles the client respones

I implemented one function at a time, first to make a sound, then to move the servo and then to draw on the display. I observed them working normally, whereas for the server I first used the serial to make sure it's connecting to the WiFi and later my laptop to connect to the server.

Optimizations

- Display Refresh – OLED updates only when lock state changes
- Single Button Toggle – one endpoint and function for toggling lock state

Demo

Resources

Software

https://www.upesy.com/blogs/tutorials/esp32-servo-motor-with-arduino-code-sg90 - servo calibration

https://randomnerdtutorials.com/esp32-servo-motor-web-server-arduino-ide/ - another servo code I studied

https://randomnerdtutorials.com/esp32-ssd1306-oled-display-arduino-ide/ - display calibration + usage

https://randomnerdtutorials.com/esp32-web-server-arduino-ide/ - client response handling

https://github.com/stefungureanu/ESP32-remote-box - repository for the code

https://www.silabs.com/developer-tools/usb-to-uart-bridge-vcp-drivers - needed to connect to the ESP32 with a USB

https://www.electronicwings.com/esp32/esp32-timer-interrupts - timer interrupt guide, used for the client timeout

Hardware

https://lastminuteengineers.com/esp32-pinout-reference/ – shows exactly which pin supports what

https://www.youtube.com/watch?v=2YbPyw5PS9A – useful video to show how ESP32 can be mounted on 2 breadboards

https://esp32io.com/tutorials/how-to-power-esp32 – very important read in case you want to power the board and the components with different sources

Export to PDF