Assignment 2: Electronics

Due 2/27/2022 at 11:59pm

In the previous assignment, we have explored the basics of Arduino and some simple electronics. We will be learning more electronics and build a simple door alarm in this assignment.

Note: You do not need to turn in the solution for every single problem. Only submit what is marked as ***Deliverables****. Combine all your answers into a single PDF in order, including the links to your videos. Submit the PDF to* bCourses.

1 Getting the interface ready

1.1 Push-button switch

Last time we looked at the potentiometer which is an analog sensor. While it is useful for continuous adjustment, for example, changing the brightness of a lamp, we often also need to change state discretely. The push-button switch is a digital sensor that does this job. Connect your components like below. You will need

- 1 × 10kΩ resistor
- 1 × push button switch

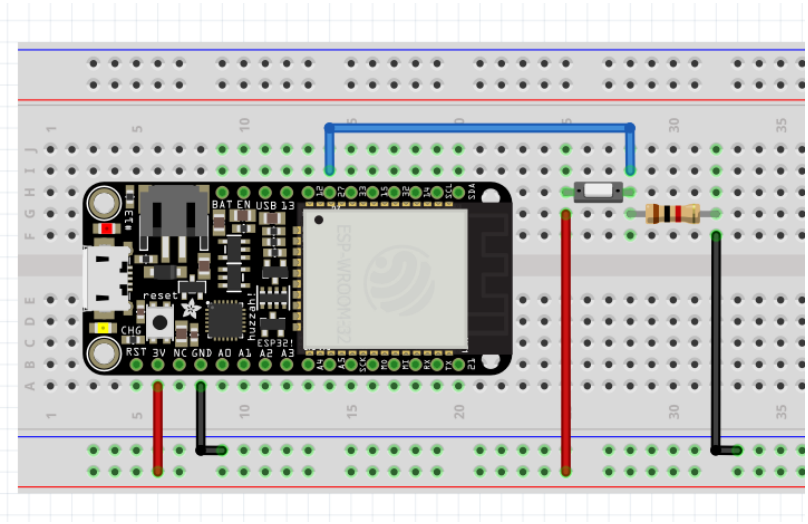

Figure 1: Push button switch circuit

Write a simple program to test if the button is working. You should set the "pinMode" of the button to be "INPUT", and use "digitalRead" to read the sensor. You should be reading "1" if you press the button and "0" otherwise. The resistor here, which makes the reading "0" by default, is called a pull-down resistor. The one making the reading "1" by default is called a pull-up resistor.

Deliverables **Written:** How do you change the resistor connection in the circuit such that the reading will be "1" by default and "0" if the button is pressed?

1.2 Interrupt

The method we have implemented so far for sensor reading is called polling, that is, the processing unit keeps requesting data from the I/O device no matter if it desires CPU processing. This is like you have an Amazon package on its way to you and you open the door every 1 minute to check if it has arrived. Sounds really dumb, doesn't it? I bet you would rather put on a doorbell and only have to go once if you hear it ringing. You can apply the same method to ESP32, and that is called an interrupt. In the case of an interrupt, the device notices the CPU that it requires its attention.

Open **push_button_switch_interrupt.ino**. This is a simplified interrupt code. Upload it to your ESP32 and open the Serial Monitor. You should be reading "Pressed!" every time you press the button.

1.3 Debouncing the push button switch

Perhaps you have noticed that multiple "Pressed!" may pop up even if you only press the button once. This is called switch bouncing. When you close a switch it tends to literally bounce upon the metal contact which connects the circuit. Usually, switches take a few microseconds to a few milliseconds to completely close.

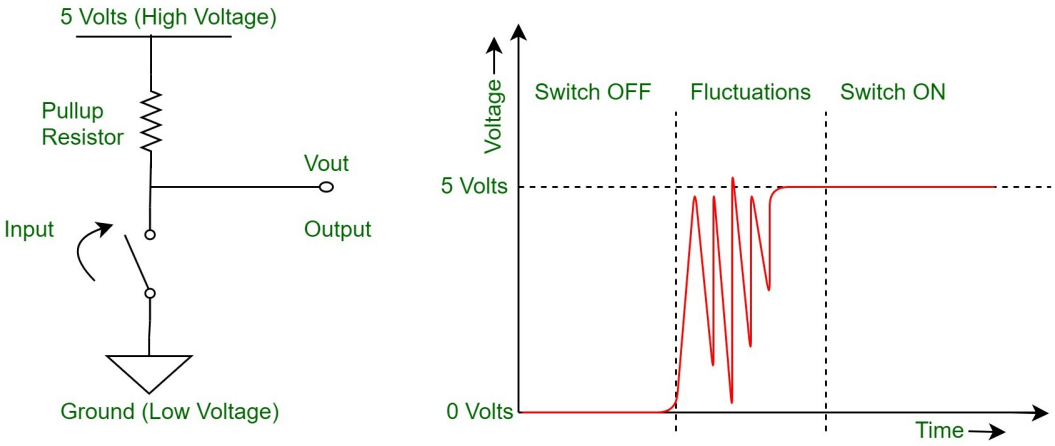

Figure 2: Switch Debounce

Modify the interrupt code such that it debounces the switch. You can use the timer (https://www.arduino.cc/reference/en/language/functions/time/millis/) in Arduino to help resolve the bouncing issue. Feel free to use any online resources.

Deliverables **Written:** Briefly explain your strategy for debouncing the switch.

1.4 Setting up the speaker

For the next step, we are going to learn how to use a speaker with ESP32. Add a speaker to your circuit. You would want to use a pin with DAC functionality, one pin that has such a feature is A1 (GPIO 25).

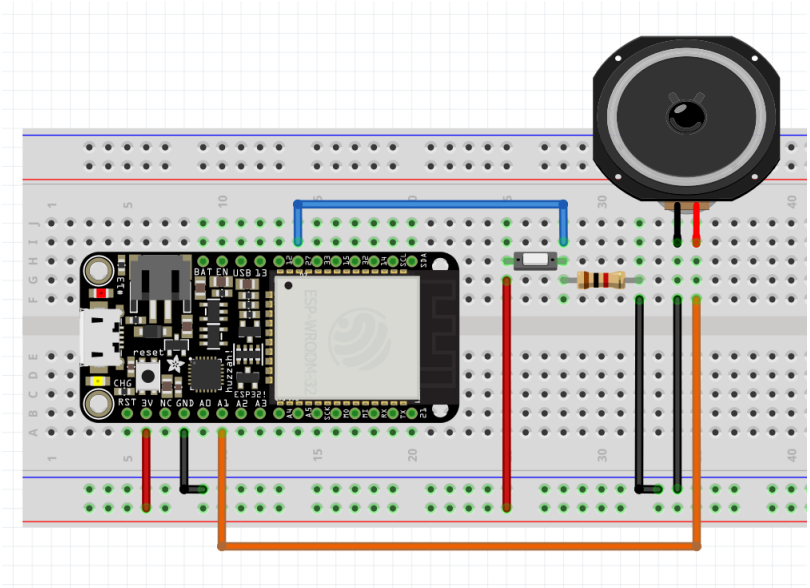

Figure 3: Speaker connection

If you have played with Arduino before, you probably know that it has a **Tone** library that allows you to play music using a buzzer. Unfortunately, this library is not present in the ESP32 environment, so we have to install it. Go to https://github.com/lbernstone/Tone32 and download it as a "zip" file. Then install the **Tone32** library by going to **Sketch ->Include Library ->Add .ZIP Library...**, and select the "zip" file you just downloaded.

Note: There are tons of Arduino libraries available online created by people all around the world. Using them wisely can save you a lot of work. Refer to https://www.arduino.cc/en/Guide/Libraries *for how to install them.*

1.5 Using the speaker

Open **speaker_template.ino**. Upload it to your ESP32, and it should play a very simple tune. We want to change the melody to something nicer to greet ourselves when we get back home. In the code, change "int melody[ ]" to whatever you want to play when the alarm system detects you arriving home. You can find more songs from here https://dragaosemchama.com/en/2019/02/songs-for-arduino/, or if you are a good musician, you can use your imagination and write a melody on your own. You can also modify "int tempo" to change the global playback speed.

Deliverables **Video:** Take a video of your ESP32 playing the melody you choose.

2 Wrapping up the system

2.1 Alarm sound

In **1.5**, we have decided on the melody to play when the system greets you. However, the door alarm system should also be able to play an alarm sound when it detects an intruder in the armed mode. To do this, create a new file and copy everything from **speaker_template.ino**. Rewrite your play music code

in a single function. Test this function by calling it in your "void setup()". Now, create an alarm sound by writing an "int melody_2[ ]". Then, create another function which is the alarm sound playing function that plays "melody_2[ ]". Make sure that your alarm sound is played for approximately 5 seconds every time you call it.

Note: You can try to write a single play function that takes the melody array as the argument. However, this can be tricky if you are new to C/C++. For time being, I recommend keeping them separate despite it may not look nice.

2.2 Ultrasonic sensor

The door alarm should be triggered when it detects a person in front of it. Most of the time, engineers use a human body infrared sensor (e.g. HC-SR501) to do this job. The one used in Etcheverry Hall that turns on the light of the classroom is probably using the same mechanism.

Figure 4: The human body IR sensor

However, since the lab kit does not contain such a sensor, we will be using the ultrasonic distance sensor. You can read the mechanism of the ultrasonic sensor here: https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6. Put the ultrasonic sensor in your circuit and open **ultrasonic_sensor.ino**. In addition to what you already have in your circuit, you will need

- 2 × 1kΩ resistor
- 1 × HC-SR04 ultrasonic sensor

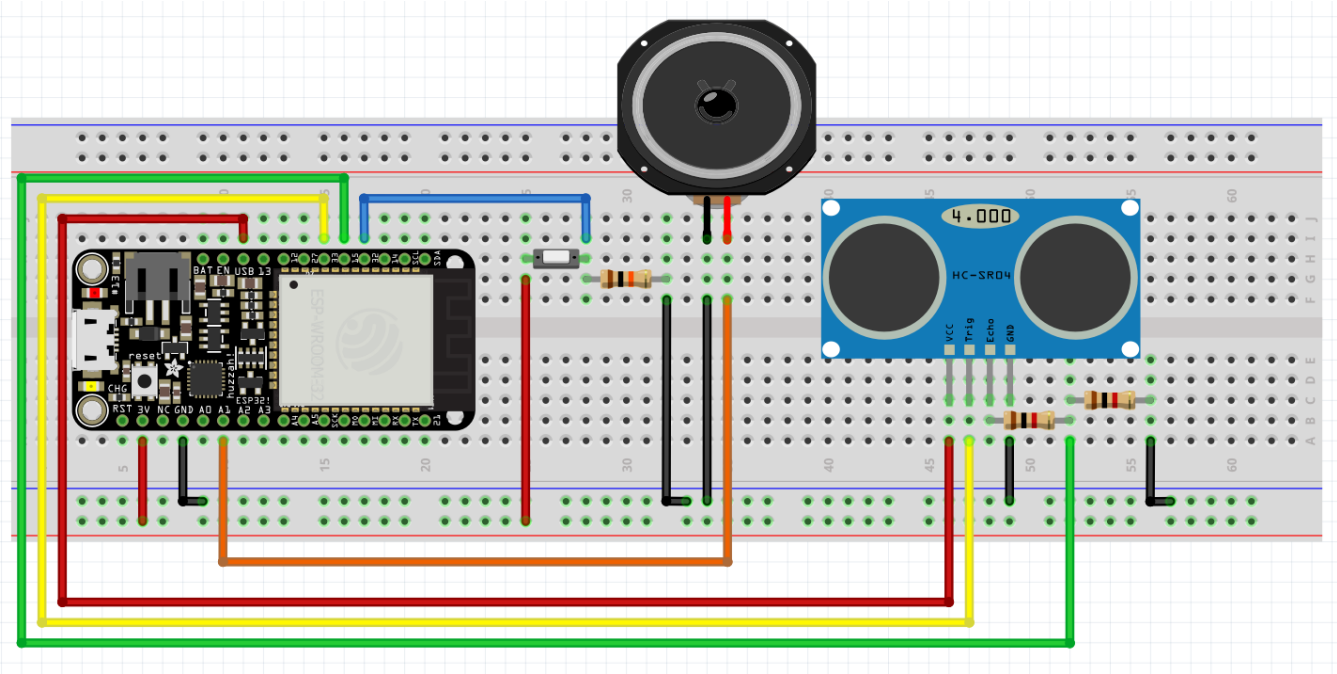

Figure 5: Full wiring diagram

You can use any pin you like, but make sure they are capable of the task you want them to do. Also, make sure there is nothing in front of the ultrasonic sensor, as it affects the reading. Besides, notice that here the ultrasonic sensor is using the USB pin as the power supply. This is because the sensor uses 5V instead of 3.3V. However, this creates a problem that the output from "Echo" is also going to be 5V. Since we know the GPIO pins of ESP32 are not 5V safe, we have to build a little voltage divider with 2 resistors. I am using 2 1kΩ resistors here. You can almost use any higher value resistor, but make sure that the two resistors you choose are identical. Upload the code to your ESP32. Open your Serial Monitor and move your hand toward and away from the ultrasonic sensor. You should be seeing the distance changing.

2.3 The door alarm system

Now our circuit is complete, and we have learned all the building blocks, we can wrap up our door alarm system. Write your own door alarm code. Overall, your door alarm system should be able to perform the following.

1. When startup, it should run in unarmed mode, where it greets you by playing the melody you used in **1.5** once it detects something close.

2. By pressing the button, it should change into the armed mode, where it plays an alarm sound for 5 seconds once it detects something close.

3. By pressing the button again, it should switch back to the unarmed mode, and the cycle continues...

Note: Put all your electronics at the back of the ultrasonic sensor and control the push button switch from the back so that the alarm will not be falsely triggered.

Deliverables **Video:** Take a video of your system performing all the above functions. You can make the melody shorter so that the video will not take a long time to film.

Helpful resources

Arduino Programming Language: https://www.arduino.cc/reference/en/
HUZZAH32 Documentation: https://learn.adafruit.com/adafruit-huzzah32-esp32-feather
HUZZAH32 Pinouts: https://learn.adafruit.com/adafruit-huzzah32-esp32-feather/pinouts
The Lab Kit: https://microkit.berkeley.edu/blank-page/
Installing Arduino Libraries: https://www.arduino.cc/en/Guide/Libraries

Deliverables checklist

Deliverables **Written:** How do you change the resistor connection in the circuit such that the reading will be "1" by default and "0" if the button is pressed? (20%)
Deliverables **Written:** Briefly explain your strategy for debouncing the switch. (20%)
Deliverables **Video:** Take a video of your ESP32 playing the melody you choose. (10%)
Deliverables **Video:** Take a video of your system performing all the above functions. You can make the melody shorter so that the video will not take a long time to film. (40%)
Assignment checkoff **(10%)** **Come to a lab session of your choice to discuss with a GSI about the assignment. (No email checkoff unless approved by GSI)**