**Lab 1 Report**

Samantha Reksosamudra - 2276717 January 19, 2024

Assignment: Getting Started with Arduino Mega

1. **Introduction**

The learning objectives in this lab included exploring the Arduino IDE and building a basic sketch using the Arduino Libraries. The sketch then is uploaded to the Arduino Mega board to observe the behavior of the built-in and wired external components (i.e. LED and buzzer). This lab also provide good practice to set up the Arduino board to a PC or laptop to establish a connection between both of them (i.e. setting the board and port connection). This lab was also a good practice to build or modify a sketch to needed to be modified in certain procedures to demonstrate different behaviors of LED blinking and making click sounds on a passive buzzer. By changing the delay values to be shorter, it is possible to make a continuous sound on the buzzer and also execute multiple tasks at around the same time (i.e. simultaneously).

1. **Methods and Techniques**

The following materials were used to complete this laboratory activity:

1. Arduino Mega Microcontroller board
2. External Arduino power supply (120VAC power adapter)
3. LEDs and 330 Ohm resistors
4. Passive Buzzer
5. USB- Type-B cable (with USB-A or USB-C for the computer end)
6. Solderless Breadboard
7. Wires

*Procedure 1:* The first procedure in this lab involved setting up and exploring how to use the Arduino Interactive Development Environment (IDE) on an Arduino MEGA board. The IDE must recognize the board used before uploading a code, so the Arduino board type and port location is specified first. Once the board is plugged to a power source (i.e. PC or laptop) via USB, there will be a steady green “ON” light indicating power. Using an example code inside the Arduino IDE library called “Blink”, the code is compiled (if there are no bugs) and uploaded to the board.

*Procedure 2:* The second procedure in this lab involved modifying the (code) sketch and LED. The rate of time of LED blinking is varied by the delay values. By decreasing the delay value, the LED will blink much faster. To observe the Arduino board’s non-volatile memory behavior, the board is unplugged from the USB cable and plugged into the 120 VAC power adapter.

*Procedure 3:* The third procedure in this lab involved using an external LED hardware to test the previous “Blink” sketch. A 330 Ohm resistor was connected between the cathode (short wire) of an LED and a GND/0V pin on the Arduino, while the long LED wire is connected to the IO/PWM pin 10. The resistor and LED connection can be done using wires or the solderless breadboard. The code was modified using #define so that the LED pin is changed to pin 10.

*Procedure 4:* The fourth procedure in this lab is a continuation of the third procedure: while the external LED is still connected to the board, a passive buzzer is wired between pin 2 and +3.3V (on the board). The sketch was modified to make click sounds on the buzzer using digitalWrite(). Similar to how to make LEDs blink, the buzzer was set to HIGH then LOW to make click sounds. To make click sounds as the LEDs change states, the external LED (pin 10) was set to ON, onboard LED (pin 13) OFF, and buzzer HIGH and do the opposite after a certain delay before looping again.

*Procedure 5:* The fifth procedure in this lab involved modifying the sketch to emit a continuous tone of 250 Hz instead of clicking sounds using if-statements and millis() function. By changing the delay value to 2ms (T=1/250/2=2ms), it will generate a 250 Hz sound, and we can determine the sound duration by keeping a timestamp using the millis() function.

*Procedure 6:* The sixth procedure in this lab involved using an oscilloscope to measure the waveform from the Arduino and ascertain that the buzzer’s frequency was 250 Hz.

**C. Experimental Results**

1. Procedure 1

Once the Arduino Mega board was plugged into the laptop via USB, there was an option to select the type of board and which port (on the laptop) the board is connected to under the Tools banner in the Arduino IDE. After the connection is established and the “Blink” code sketch is opened successfully in the IDE, the sketch is compiled and uploaded to the Arduino board. The result was the onboard LED (on the Arduino board) was blinking one time every two seconds.

1. Procedure 2

After changing the delay values from 1000ms to 200ms, the sketch is recompiled and re-uploaded to the board. This resulted in the LED blinking faster than in Procedure 1. After unplugging the Arduino board from the USB cable, the board (and onboard LED) was turned off. But when plugged again into another power source (using the 120 VAC power adapter), the board turned on again and the LED was blinking again. So the blinking light program was automatically loaded on the board due to the Arduino’s non-volatile memory.

1. Procedure 3

A resistor was connected to the GND/0V pin on the Arduino and a breadboard, where it’s connected to the cathode of a red LED. The anode of the LED is wired to the IO/PWM pin 10. The modified sketch included a line of code that defines LED\_EXTERNAL as pin 10 (instead of pin 13) using #define. After the modified sketch is recompiled and re-uploaded, instead of the onboard LED, the external red LED on pin 10 blinked at the same rate as the onboard LED in the previous procedure.

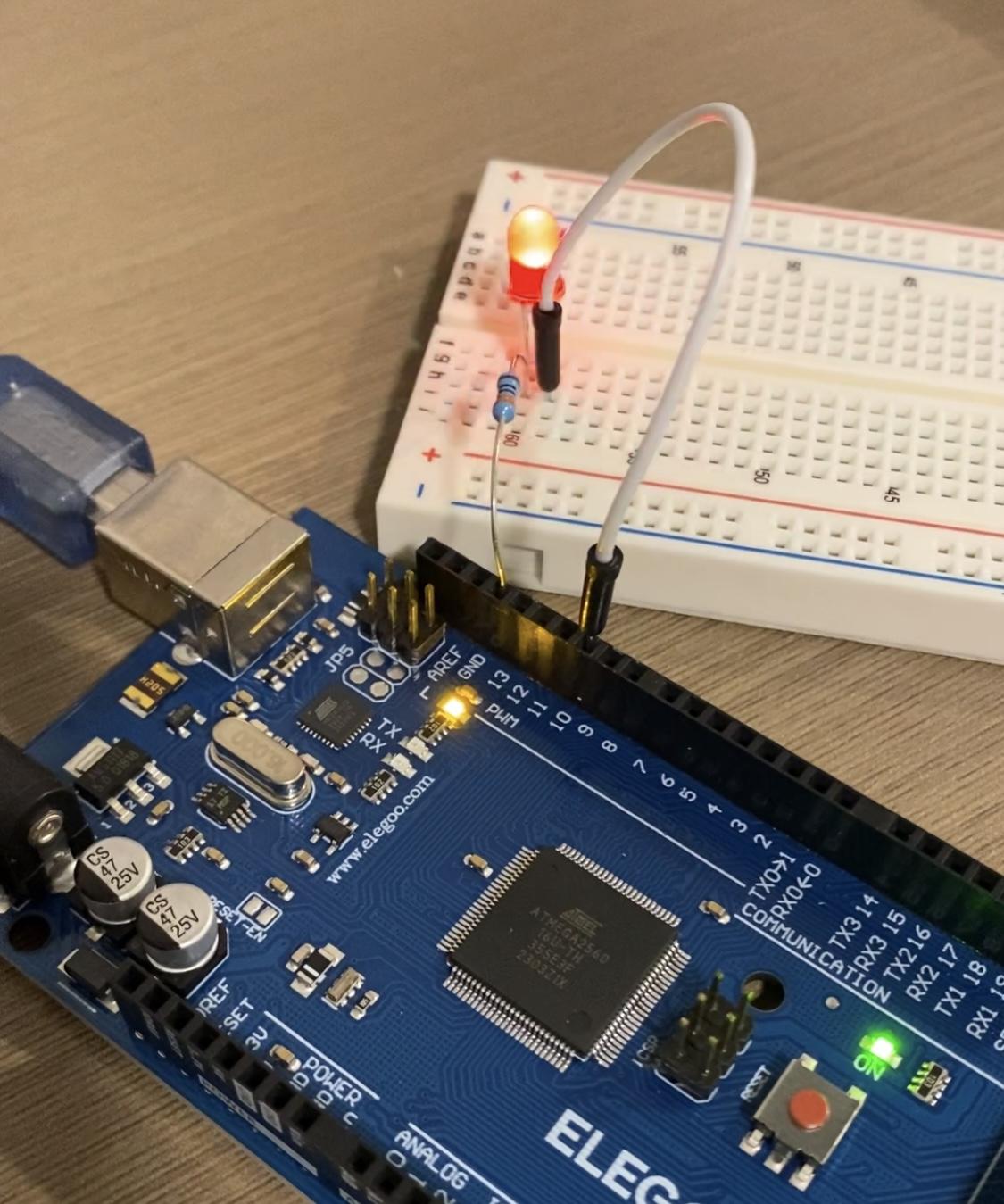


Fig.1 LED and resistor setup

1. Procedure 4

After modifying the sketch so that it uses digitalWrite() on the buzzer instead of the LED, the sketch writes HIGH and LOW values to the buzzer after a certain amount of delay (i.e. 200ms). So the click sounds were as fast as how the LEDs were blinking. Then after using the other sketch modification, the external LED (pin 10) and onboard LED (pin 13) were switching between one another from ON and OFF state, and the buzzer clicks when the LEDs change state.

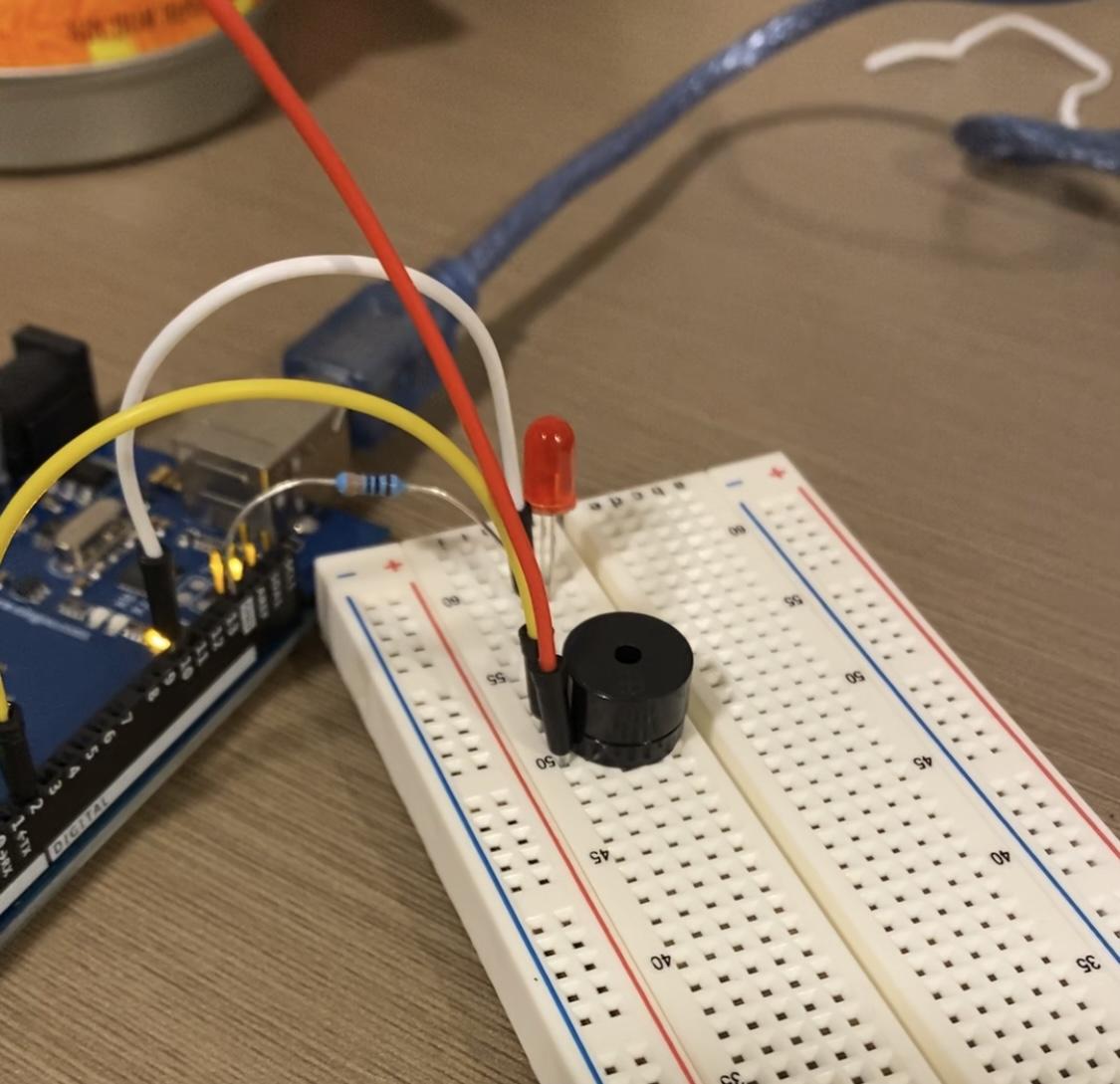


Fig.2 Buzzer is connected to pin 2 and 3.3.V

1. Procedure 5

By calculating the period and delay time needed to generate a 250 Hz tone, the modified sketch was recompiled and re-uploaded to the board. It resulted in the buzzer emitting a continuous sound for 3 seconds (since I set duration=3000), but the LEDs kept flashing the same way as in Procedure 4.

1. Procedure 6

Using an oscilloscope, a wire is attached to the buzzer and connected to the oscilloscope to display a waveform and the frequency of the buzzer. However, the waveform needed to be adjusted first since the waveforms were too close to each other. The horizontal knob controls the time/division of the screen, so turning it to the left will show a clearer rectangular-shaped waveform. While the vertical knob controls the voltage measurement of the signal, so tuning it to the left or right will correct the vertical length of the rectangular-shaped waveforms on the screen. After some tuning, the result showed a stable waveform and the frequency was around 200Hz. It was not perfectly 250 Hz since my breadboard had issues.

**D. Code Documentation**

1. Procedure 1

The sketch always has a void setup() and void loop() function. The setup function runs once when the board is resetted or connected to a power source. Figure 2 shows the “Blink” sketch where the setup function initialized the digital pin for the onboard LED as output and the loop function used digitalWrite() to write high and low values to the LED after 1 second each time. The sketch used milliseconds (ms) instead of seconds (s), so the delay value was 1000.

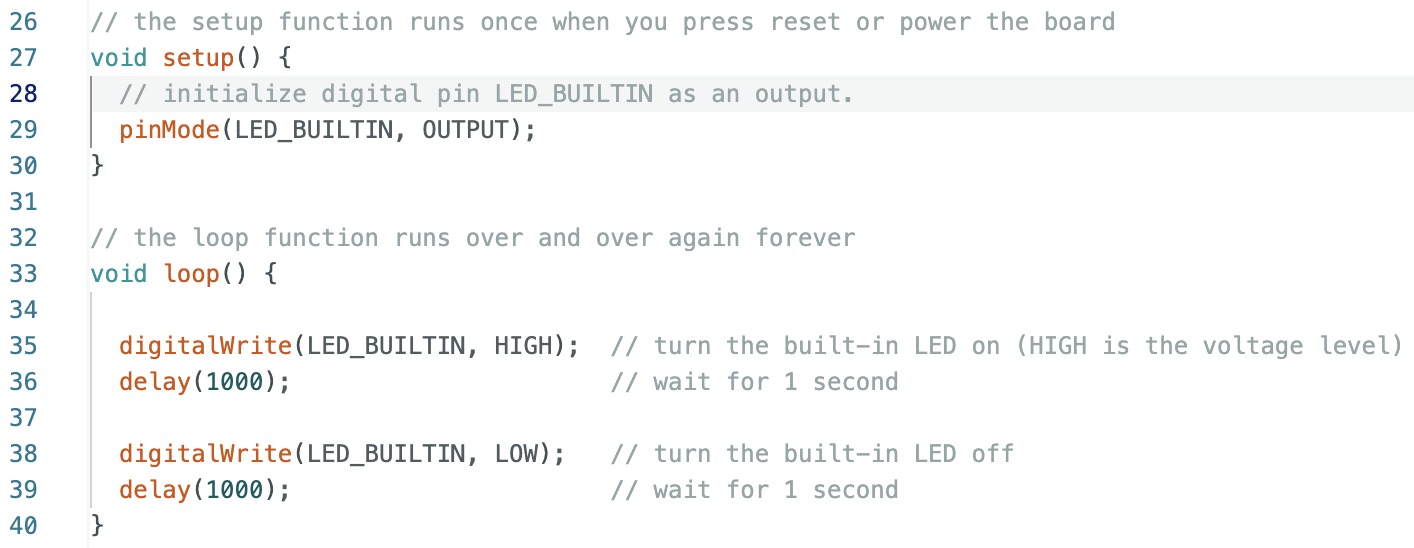


Fig.2 Procedure 1 sketch

1. Procedure 2

Instead of delaying the LED by 1 second, the sketch below delayed the LED by 0.2s or 200ms. So the LED was on for 0.2 seconds before turning off for another 0.2 seconds. And the loop repeats.

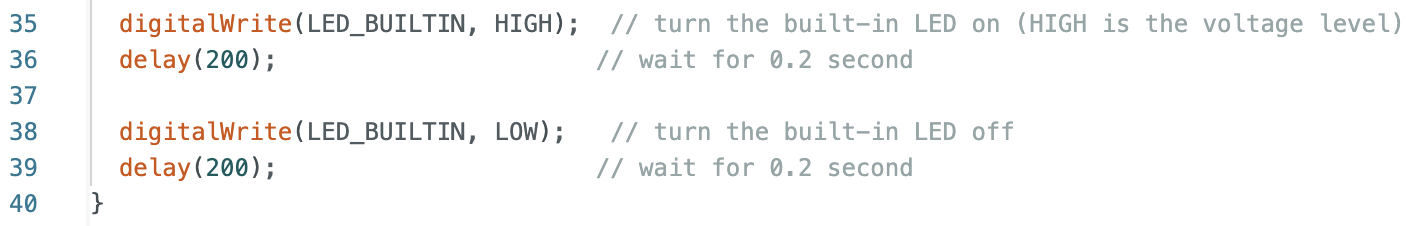


Fig.3 Procedure 2 sketch

1. Procedure 3

By using #define, the sketch below sets all variables called “LED\_EXTERNAL” into (digital pin) 10. This resulted in the sketch controlling the external LED in pin 10, instead of the onboard LED. If the external LED was changed to connect to another pin (i.e. pin 8), then the only thing needed to change is the pin number in line code 26.

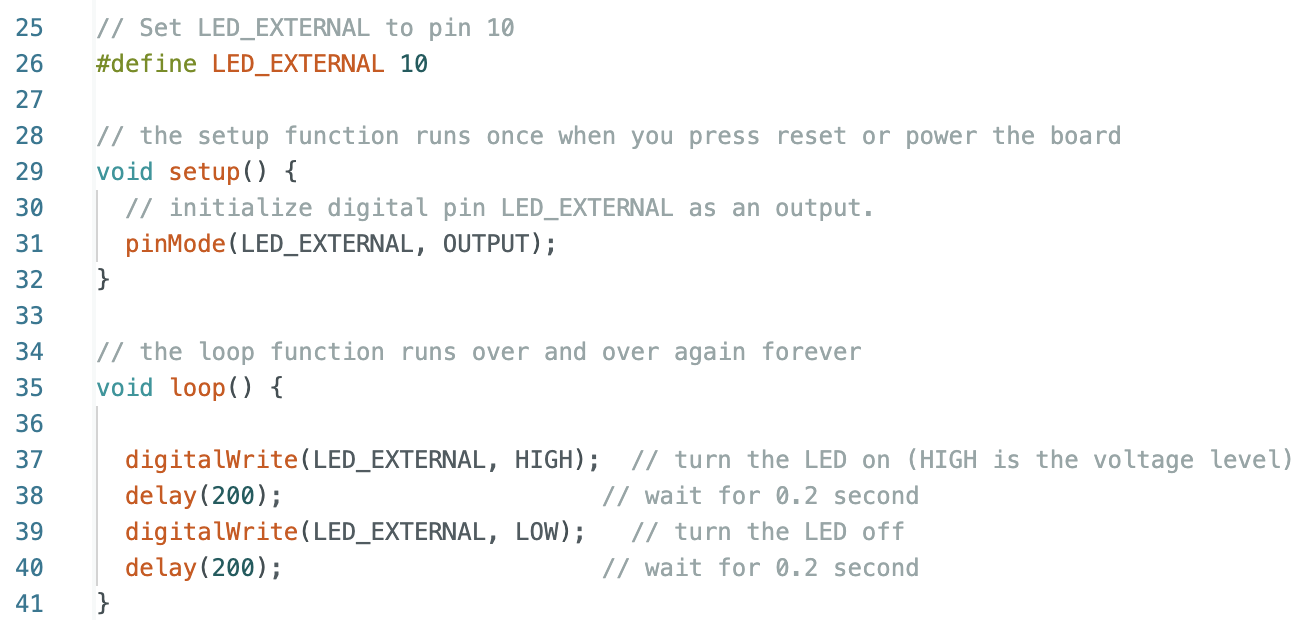


Fig.4 Procedure 3 sketch

1. Procedure 4
   1. Generate click sounds

Similar to the sketch in Procedure 3, the sketch defined the buzzer in pin 2 and initialized it as an output in the setup function. And the loop function writes high and low values to the buzzer in interval times.



Fig.5 Procedure 4a sketch

* 1. Generate click sounds when LEDs change state

The sketch was modified so there are click sounds whenever the LEDs change state. So there are two #define statements for external LED and buzzer. The built-in LED does not need a #define statement since it is automatically initialized as pin 13.

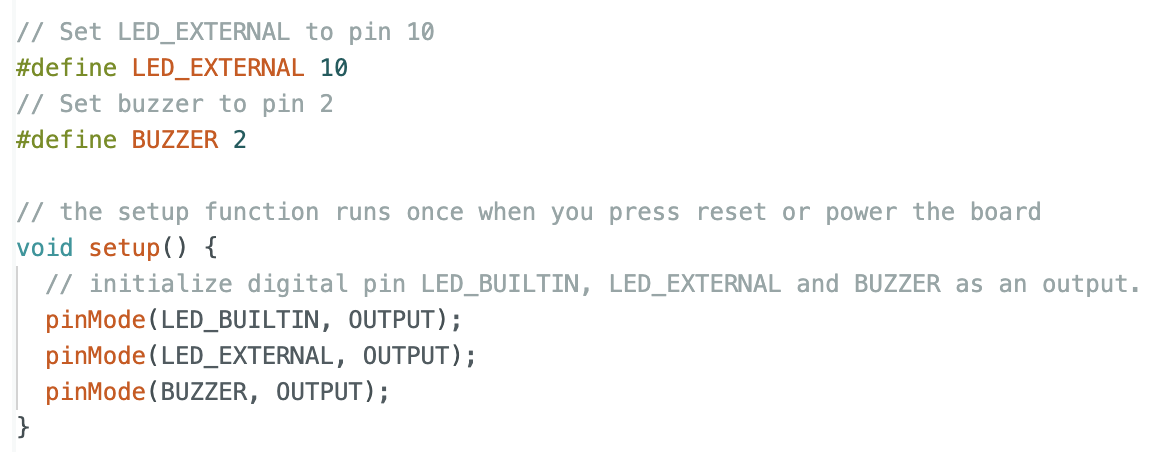


Fig.6 Procedure 4b sketch (define statement and setup function)

In the loop function, the built-in LED was on first while the external LED was off, and the buzzer was set on high (so it makes a sound). After 0.2 seconds, the LEDs will switch state, and the buzzer will be turned off (which will make another click sound).

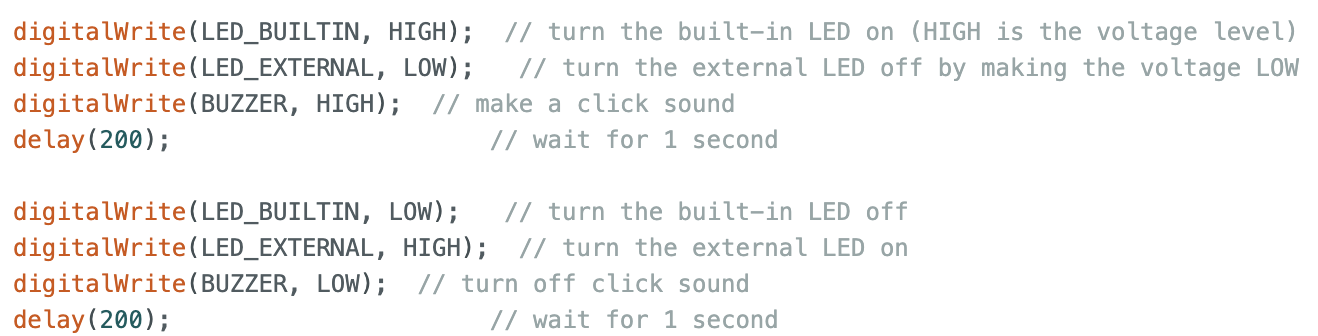


Fig.7 Procedure 4b sketch (loop function)

1. Procedure 5

The figure below shows the variables used in Procedure 5: FREQ defines the frequency value of the sound, interval is the delay duration for LEDs, duration is the how long the sound plays before it turns off, prevTime is the timestamp of when the LEDs change state in the previous state, and startTime is the timestamp of when the setup function started running.

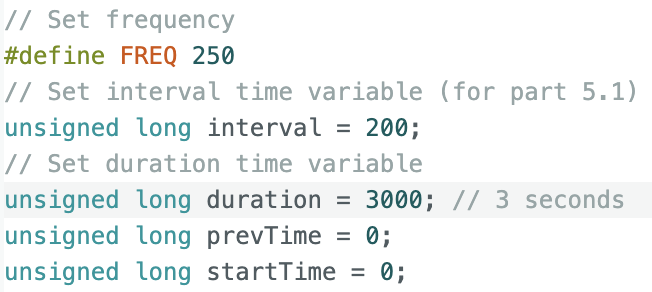


Fig.8 Procedure 5 variables setup

The figure below shows the values assigned to prevTime and startTime when the setup function ran. The millis function returns the current timestamp of when the line of code is run. And the initial LED state was set that the built-in LED is turned on first and external LED is off.

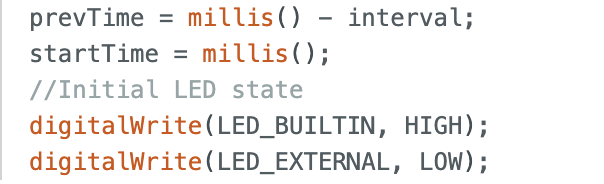


Fig.9 Procedure 5 setup function

The loop function had two if-statements that handled the switching LED states, and played sound on the buzzer. The first if-statement will run at each timestamp after a specific interval (i.e. if interval is 200ms, then the code will run every 200ms). The code inside writes the opposite value of the current state the LEDs are in. While the second if-statement will run as long as the current timestamp is less than the duration (i.e. if duration is 3000ms, then the code will run for 3 seconds). The delay is set to 2ms, because the period of a 250 Hz signal is 2ms.



Fig.10 Procedure 5 setup function

**E. Overall Performance Summary**

During the demo session, the Arduino Mega board was successfully connected to the laptop and performed the Blink sketch, which blinks the onboard LED once every two seconds. By modifying the sketch code, the board can make the blinking go faster, change which LEDs it controls, and make click sounds on a passive buzzer. Through some modifications, the board also successfully played a continuous tone of 250 Hz while blinking the on-board and external LED. However, during the oscilloscope demo, the signal did not display a clean rectangular-waveform. There were a lot of noise even after the tuning process. After using a different code (that was sure to work), it was found that the breadboard was the issue because it was not grounded.

**F. Teamwork breakdown (not applicable to this report)**

**G. Discussion and conclusions**

The challenging part of this lab was figuring out how to generate a continuous tone while blinking the LEDs simultaneously. Since C programming executes a line of code sequentially (i.e. not possible to execute multiple tasks simultaneously), it took some discussion with TA to discover that delays in C Arduino code is very fast and might seem like there was no delay at all. And both tasks were executed simultaneously. I was especially proud of the sketch for Procedure 5 where I learned how to use the millis() function and digitalRead() to make the code more efficient. I have learned how to use digitalRead() before but the millis() function was something new that I learned through this lab.