Getting Started with the Arduino Mega

Anna Mai (2165101)
06/30/23

Assignment: Lab 1

Introduction

This lab is an introduction to the development process when working with an Arduino Mega and other similar boards. It offers guidance starting from the Arduino IDE (Integrated Development Environment) to building and executing programs on an Arduino to manipulate devices.

The lab is separated into 6 parts. Part 1 focuses on the Arduino IDE, and uses a given example sketch called “Blink”. Students are shown through compiler errors as well as how to configure the IDE in order to upload the sketch onto an Arduino board. Parts 2 through 5 focus on modifying the example sketch in order to change the board output. Part 6 explores the functionality of an oscilloscope as well as how to use it to analyze a program’s output to a board.

Methods and Techniques

Part 1 of the lab is centered on the first learning objective, teaching how to set up the IDE as well as how to navigate through its UI. It shows how to see if your program has an error as well as how to upload an error-free program to a connected board.

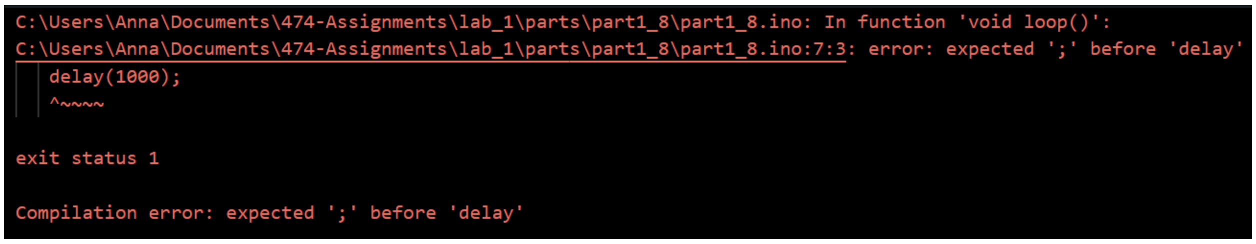
Parts 2 through 5 are centered around the second learning objective, which involves building and uploading programs onto a board. In part 2, students observe the code structure and understand its purpose and how it controls the onboard LED. In part 3, students learn how to connect external devices to the board and how they can be controlled, by modifying blink.ino to a different pin number output. Students also learnt how to properly set up a LED, using a resistor to limit the current, so that the LED doesn’t burn out. In part 4, students learn how the same program can be used for a completely different function by changing the device output. By changing the device the board controls from an LED to a speaker, the program now outputs audible clicks rather than visible blinks like in previous parts. In part 5, students learn how to manipulate different pins at different frequencies simultaneously.

Lastly, part 6 shows students how to measure signals with an oscilloscope, as well as how to use the dials on the oscilloscope to change its viewing window to better see the output signal.

Experimental Results

Part 1

Removing any semicolon will cause the code verification to fail.



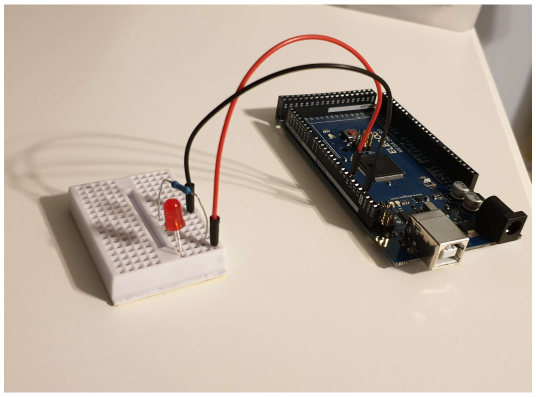
```
C:\Users\Anna\Documents\474-Assignments\lab_1\parts\part1_8\part1_8.ino: In function 'void loop()':
C:\Users\Anna\Documents\474-Assignments\lab_1\parts\part1_8\part1_8.ino:7:3: error: expected ';' before 'delay'
   delay(1000);
   ^~~~~

exit status 1

Compilation error: expected ';' before 'delay'
```

Part 2

Changing the delay value from 1000ms to 200ms makes the LED blink five times as fast.

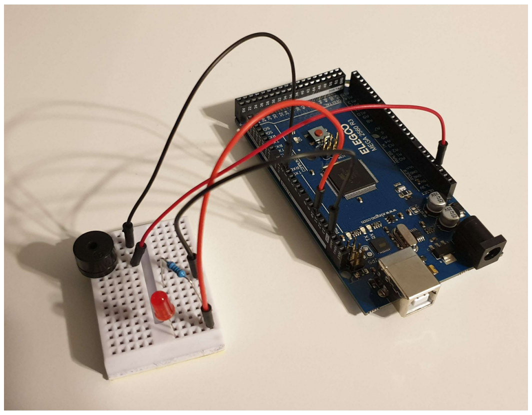
Part 3

Changing the first parameter in pinMode allows the program to control an external LED wired up to the arduino, rather than the onboard LED.



Part 4

Using the same code, but connected to a speaker rather than an LED causes clicking to occur with the same timing as the LED changing state.



Part 5
In order for the speaker to emit a continuous tone at 250hz, the speaker has to change state every 2ms. However, the LEDs are still flickering every 200ms, so every 2.5hz. For these devices to be controlled at different frequencies, a different method rather than delay was employed.

Part 6
Connecting the oscilloscope to the 2 pins controlling the speaker output results in a frequency of 246hz, the ideal would be 250hz, meaning that there is a percent error of 1.6%.

Code Documentation

setup() - lines 20 to 24
Initializes the output pins for both the onboard and external LED as well as the speaker.

loop() - lines 26 to 39
Contains the logic to control the speaker output, which buzzes at a frequency of 250hz as long as the time the program has been on hasn't exceeded the desired buzzing duration.

To achieve a frequency of 250hz, the buzzer must have a period of 4ms. The function checks the current milliseconds elapsed and determines whether or not to change the pin state to HIGH or LOW. When the milliseconds elapsed is a multiple of 4, the signal to the speaker will be HIGH; if the milliseconds elapsed is 2 greater than a multiple of 4, the signal to the speaker will be LOW. This results in a frequency of 250hz with a duty cycle of 50%.

alternateLeds() - lines 47 to 58
Contains the logic to control the LED output, which flickers every 200ms. In order to achieve this, a similar method to the speakers was used, but rather than checking for if the milliseconds elapsed is a multiple of 4, it checks if it's 400.

Overall Performance Summary

I didn't really have any issues during the demo. Prior to my demo I was worried about how my oscilloscope measurement wasn't exactly 250ms, but rather 246ms, but after I realized that it was only a percent error of 1.6% I went for the demo.

I feel confident with all three learning objectives of lab 1. I can navigate the Arduino IDE, build and run a program through the IDE as well as how to control LEDs and speakers.

Discussion & Conclusions

The only notable difficulty I encountered was controlling the LEDs and speakers at different frequencies simultaneously for part 5. Originally, I tried to modify the delay values and loop it, but I found that to be difficult to manage. Luckily I found out about milis(), which outputs how many milliseconds have elapsed since the program started running. I used that in conjunction with modulus to come to a relatively clean solution.