07/09/2020

Lab REPORT WEEK 3

# iNTRODUCTION

During this week’s lab, we get introduced to more components on the Arduino breadboard, the mechanism of these components, and how these components can be connected to build a correlation through our designed program. We learn about this by connecting the components in the circuit and write a code to get the desired effect out the components. We also get to learn how to make use of two components at the same time.

## Materials

### Buzzer:

### A buzzer processes input from the Arduino code and produces output in the form of sound. They can either be directly connected to voltage outlet or with a resistor. A resistor will make it sound quieter. To make the buzzer work, we write a certain level of Hertz to the buzzer. The higher the Hertz, the higher the pitch of sound it produces. The range of Hertz a buzzer can process is roughly up to 2048 Hz. Buzzers are widely used in real life scenarios, for applications such as alarms.

### Light sensor:

### A light sensor detects light its surroundings and records the level of brightness by outputting a number to the Arduino monitor. The bigger the reading, the brighter the surrounding environment. The reading ranges from 0~1023. A light sensor is connected to a series circuit with a resistor, with two ends connected to ground and 5V outlet. For real life application, it is widely used on the screens of smart devices to detect the environmental brightness in order to adjust the screen brightness accordingly.

# LAB TASKS

## Task 1 - Buzzer

## Pseudocode:

## Assign buzzer to pin 9

## Set frequency value to 1000 (Hz)

## Set duration value to 2000 (Ms)

## Set frequency and duration of the buzzer through pin 9

## Delay for 5000 (Ms)

## A circuit board Description automatically generated

## The Arduino code sends instruction to the buzzer through tone() function. The frequency decides the pitch of the buzzer and the duration decides how long the buzzing lasts. It is worth noting that the duration value of the buzzer has a separate timer to the main timer. This means the duration does not accumulate with the delay, but instead it goes off in parallel with the delay function. Our group was struggling on that, but as soon as we figured it out we were able to manipulate the duration and delay time in our preference. It has also come to our attention that when we tried to connect the buzzer with a resistor in the circuit, the sound of the buzzer went softer. It was apparent that the higher the voltage that goes through the buzzer, the louder the buzzer. We prefer the buzzer to be louder, and thus we did not choose to put a resistor in our circuit.

## Task 2 – Light Sensor

## Pseudocode:

## Initialize serial communication at 9600 bits per second

## Reads the input on analog pin A0

## Print out the value read from A0

## If A0 value is less than 10, print out “dark”

## If A0 value is between 10 and 200, print out “dim”

## If A0 value is between 200 and 500, print out “light”

## If A0 value is between 500 and 800, print out “bright”

## If A0 value is greater than 800, print out “very bright”

## Delay for 500 (Ms)

## A circuit board Description automatically generated

## Our code allows us to read the input from the light sensor from pin A0, which is the value detected by the light sensor. When the surroundings of the light sensor get lighter, the value gets higher with a maximum value of 1023. Our code is able to catch that input value and print it out on the Arduino monitor. When we put our hand around the light sensor, the input value from it went down, which meant the environment got darker. And then we are able to precisely identify the level of brightness using if-statement within our code. We took a bit of time to decide which range of values would be counted as bright. But overall it was simple connection and simple coding, and our group did not encounter any difficulties.

## Task 3

## Pseudocode

## Initialize serial communication at 9600 bits per second

## Reads the input on analog pin A0

## Invert and multiply the value from A0 to get a corresponding frequency value, so that the lower the light sensor reading, the higher the frequency

## Print out analog reading and frequency value respectively

## Assign buzzer to pin 9

## Set buzzer frequency to the frequency value converted from analog reading

## Delay for 500 (Ms)

## A circuit board Description automatically generated

## As shown in the image, we are connecting both the buzzer and the light sensor to the breadboard circuit. However, they are not in the same circuit. They act as individual parts in different circuits, and their functionalities are only combined with our code. Since one output value to the computer and the other reads input given by the code, we are able to convert the light sensor reading into a corresponding frequency value for the buzzer, so that the frequency of the buzzer changes accordingly to the light sensor readings. We did not set a duration value to the buzzer, and therefore the buzzer continuously buzzed, allowing us to hear and track the changes of the light more clearly. The delay value is set to 500, which means every half a second we update the light sensor readings, and thus the buzzer frequency as well. Our group member had trouble with understanding how the two components were related and affected each other. With a bit more explanation, everyone eventually understood that the code was the one thing drawing connections. This task showed us how to make use of two components on the breadboard at the same time, without having to physically connect them in the same circuit.

# RELATION TO REAL-WORLD ELECTRONICs

## Application Idea:

## There are great potentials for the utilization of the components that we used in this week’s lab. For example, we can use them both in an old-fashioned wall clock for a build-in buzzing system. The buzzer can be set to buzz every time when the clock is on the hour. However, the light sensor can detect when the surroundings gets darker, which means it is sleeping time, and shuts the buzzer down. The code loops every hour.

## Connection:

## A circuit board Description automatically generated

## For this idea, essentially, we can use the same connection as in task 3, which is to connect the two components separately to Arduino and draw connection with coding.

## Pseudocode

## Initialize serial communication at 9600 bits per second

## Reads input from the clock itself for the time in the unit of o’clock

## Reads the input on analog pin A0

## If it is dark, do not proceed to any action

## If it is light, assign buzzer to pin 9

## buzzer goes on for 2 seconds in a medium frequency

## Delay for 3600000Ms

## Risks and Difficulties

## One of the challenges of this system will be to retrieve information about the current time. As this system is not connected to an actual clock, the buzzer might not go off exactly on the hour. This means the only way to get it work is to manually turn on the buzzer exactly on the hour. This is not very practical in real-world scenarios.

## It is also very important to make sure that the light sensor functions properly. In the scenario that there is a defect in the light sensor, it is possible for the buzzer to continue going off every hour, regardless of the changes in environment. This simple malfunction could potentially abort the whole system. It will be better to also test the light sensor to make sure it is working, and turn off the buzzer if any issue occurs, in order to save effort for the user to physically shut it down.