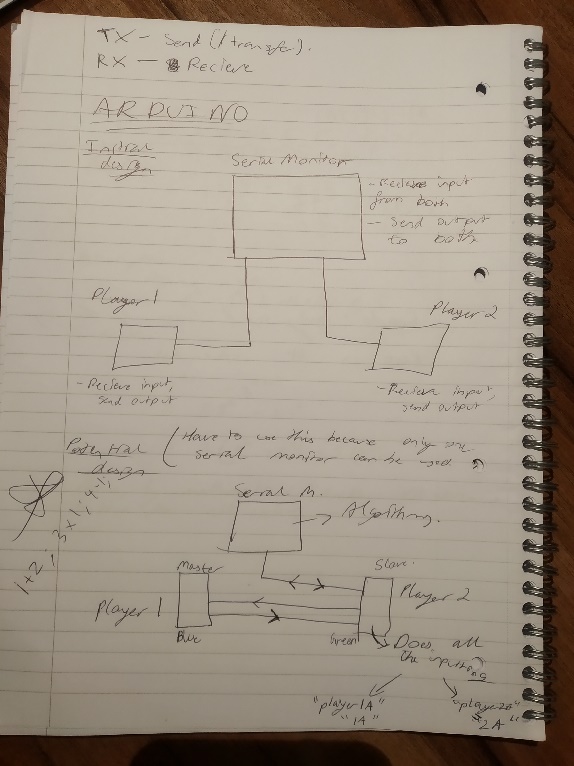
Math Quizzer

CW2 GROUP 09| Computer Architecture | 09/12/2019

# Introduction

Being tasked with the objective of creating a project that requires two Arduinos to communicate with each other in order to accomplish a goal, we decided to create a quiz game that would not only be fun but also beneficial to those who would potentially be using the Arduinos. It also meant that we had to think about how to generate questions and work backwards and forwards through the way that the program had to be built. It would be an interesting project in that both the technical challenge of implementing different types of questions efficiently would be crucial to the success of the program as well as the communication aspect of the task set.

# How project was implemented

What we understood was that there would be three components- a serial monitor, and the two Arduinos and the three had to be in/directly linked. Initially it would have made sense to take the first approach shown in the image here, with the Arduinos being linked by the Serial monitor so that they could share the monitor in between them and the timing would also correlate. However, we found that with doing this, it would also mean that they would have separate serial monitors rather than a common shared monitor; this would have made the whole program rather messy and unclean.

Instead we decided to go for the second approach that at first appears to be more complex than the first but can be implemented more easily. The slave Arduino code contains the code for the equation generation and the functionality that allows it to operate the piezo, servo, LEDs and check for the Arduino pressed first as well as who was correct out of the two. We also made sure that we did not implement interrupts but rather to manually check with each loop whether a button had been pressed, with the intention of making the functions easier to write. Also, it meant that we were not limited by having to use only the two given interrupt pins and could use any and all of them for the buttons. Hence, no shift register had to be used as we initially thought, and code was simplified.

Looking inwards, the program consists mostly of independent functions that can easily, with a few modifications, work well within any given program. Each of the functions that generate an equation to solve or a set of equations can independently generate them, print them to the serial monitor, print the options, and return the correct option as an integer out of those options. The order of the type of questions is always the same every time the question is run for a fair distribution of points. The rest of the program works in the form of loops and if conditions to check whether a button is pressed and to decide what to do.

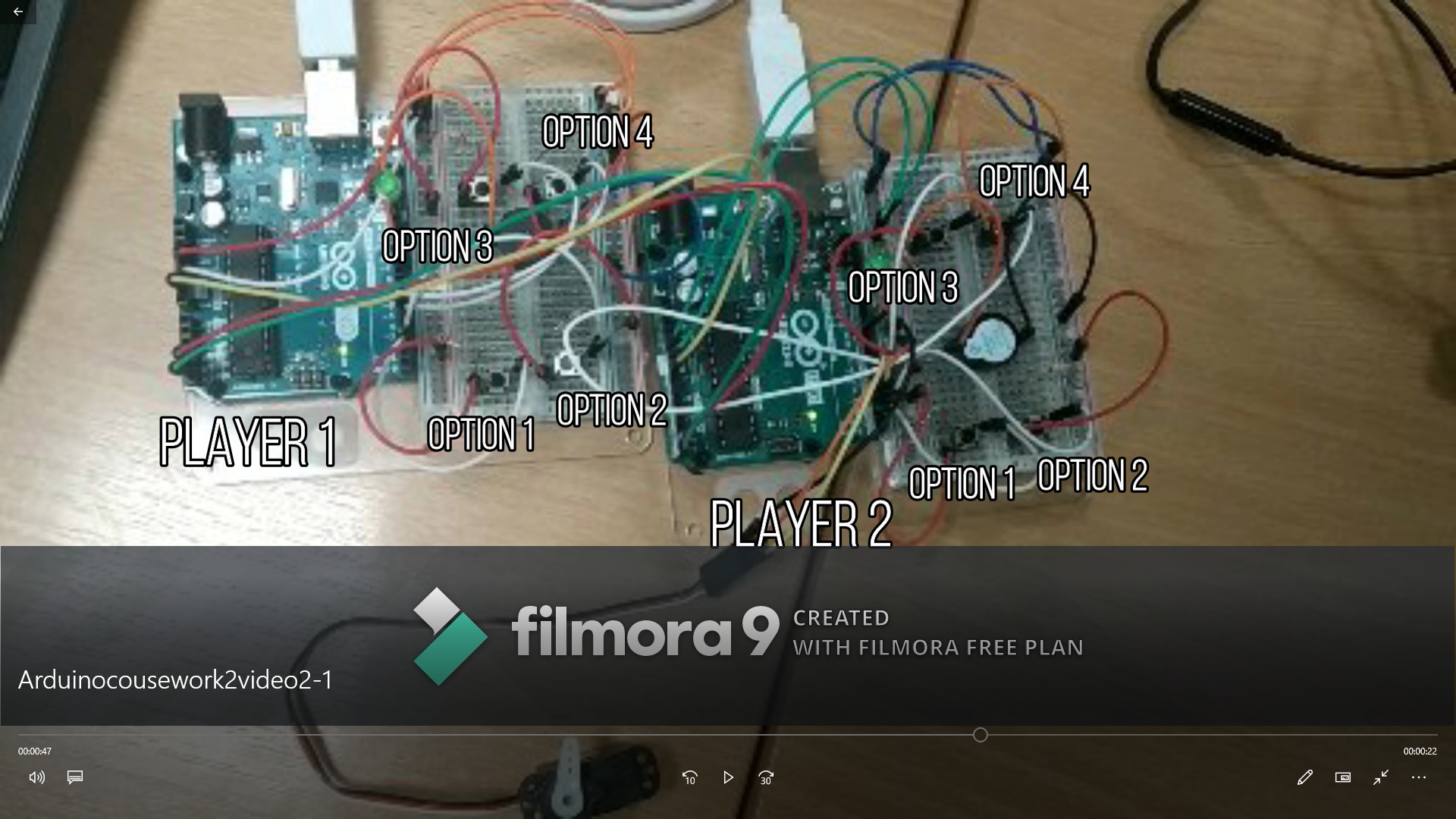
# How it works

The program first starts off by importing the necessary libraries to operate the servo and communication libraries. All the global variables are declared and the functions to generate random numbers also declared. Global variables are used to save on storage space and convenience since they can be accessed from any location and modified accordingly. The first function starts off with generating three random numbers by creating an x coefficient term and two constants. The function then calculates the root by subtracting the first constant from the second constant and then dividing by the x coefficient term. Every single element in the array is assigned the value of ‘0’ so that when the program wants to insert random values into the options, then it checks for whether the option is ‘0’ to replace it- otherwise it rolls the die again and it finds another place to slot in the number.

For quadratic equations, roughly the same events occur in the same order. All variables are given the value of 0 and then inside of a while loop with the condition of having all the variables so that they give the determinant as greater than 1, assigning values that randomise them as long as they give a valid determinant. The program will then print the question. The program then finds the determinant separately to doing the quadratic formula to find the answers because of the complex calculations that have to be made and the brackets involved. However this time a two dimensional array is created to store the options since the solutions always come in pairs. A random value for the position that the answers have to be in is generated and the pair of solutions are put in the corresponding slot in the options. One fundamental was realized when coding this was that when the other options are generated, not only do they need to be in random positions, but also their values: if the correct answer is always the answer which is either the biggest, the smallest or a middle value then the program becomes very predictable and useless. Hence the reason why the function to generate either one or a zero had to be created to decide whether randomly generated numbers are bigger or smaller than the correct root.

Simultaneous equations are created by working backwards from the solutions to make sure they give the correct answer when worked out. The rapid fire questions are made by using a 4x4 array and storing answers in a single array. The slave code also has a function called receiveEvent that executes the function to check which player answered first and prints which answer was output first. The slave acts as the middle man between the serial monitor and the master Arduino and does all of the processing and hence contains all of the code too.

# Appendix



A close up of a map

Description automatically generated