**CPEN 291   
Mini-Project Report**

**A. Group info**

Lab section: L2A (*L2A or L2B)* Group #: A-G4 (*Example: A-G1*)   
Group’s Lab Bench #s: 9 and 4 (*Example: 1 and 2*)   
Student names:

|  |  |
| --- | --- |
| Andrea Shao | Brielle Law |
| Matthew Chow | Matthew Stefansson |
| Matthew Yen | Sebastian Gonzalez |

All the underlined text in this template are for your info. You should remove any text that is underlined before submission to Canvas.

Please keep the format of the report as is (e.g. do not omit any section, or change the font size or margins).

In Sections B and C of the report:

* Explain the design and implementation procedures, and thoroughly provide documentation for the circuitry and software. During the project, you must have selected any method over another for some steps, describe the alternative (e.g. the second best) you considered. Include block diagrams or drawing to identify the main components and their interactions.
* You may include code segments in this part of the report only whenever needed for the explanations of the software design and approach. Your code must include comment statements, so do not repeat what is already included in the comment statements. As usual you will need to submit the complete code file separately, and also to include the complete code as an appendix to this report. The code must be readable in the first place and include sufficient comments (per code segment and per line, when needed) for documentation.

**B. Technical documentation for the main functionality**

Fully document your design and implementation for the main functionality here. In particular explain:

* What the six dance moves are, and how they are different
  + Dance 1: The robot taps its left foot a few times, and then shakes its right foot. It then taps its right foot a few times and then shakes its left foot. This movement is then repeated, rotating between the two feet.
  + Dance 2: The robot points its feet outwards and then inwards in a quick sweep before beginning to tap and alternate between its two feet. It taps for a total of 6 times before repeating the sweep and tapping cycle.
  + Dance 3: The robot initially has its feet facing outward and slightly upward. It then quickly turns its feet in towards the center of its body and changes the angle of its feet. By doing this, it jumps momentarily off the ground. It then quickly shifts its feet back to the starting position.
  + Dance 4: The robot raises its left and right foot in sync. The feet will tilt right and left, slightly raising the robot. Next, the legs will rotate in half circles, moving the feet outwards then inwards. The robot repeats this movement at different speeds. After this, it will return to the starting position.
  + Dance 5: The two feet both shift right then left and repeats this shifting movement. As it shifts its feet, it also moves backwards at the same time. Overall, this dance move appears like a zig zag backwards shuffle.
  + Dance 6: The robot shifts the bottom motors to make the robot look like its standing on its tippy toes. To do this, it rotates its feet inwards changes the angle of the feet dramatically so its raised high off the group. It then returns to its original position and repeats this movement.
  + Ending pose: The robot’s dances end in the splits position.
* How you have implemented each of the six robot moves

To implement the dance moves, each move was developed in different files (dance1, dance 2 etc). All the separate files were thoroughly tested and adjusted as necessary. Once all the subfiles were completed, they were integrated into the main coding file of the robot dancing. For a smooth process and to make the robot “dance” more naturally, we added the threading with the piezo buzzer and ultrasonic sensor.

* What info the LCD displays and how it has been mounted/attached for optimal viewing

The LCD display will show and add certain elements to our dancing robot. The LCD is mounted on the breadboard (which sits inside the robot’s shell) so that it can easily be viewed. When the robot is first turned on, an opening image of a brain and the words “Hi, I’m bub” are shown. As the robot shuffles through its different dance moves, the appropriate dance name appears, as well as the correct pose. The introduction photo and a few sample dance photos can be seen below.

A close up of a logo

Description automatically generatedA close up of a logo

Description automatically generatedA close up of a logo

Description automatically generated

**C. Technical documentation for the additional functionality**

Fully document your design and implementation for the additional functionality here. Explain:

* What the additional functionalities are
  + We added the piezo buzzer to play a song while our robot is dancing
  + We added the HC-SR04 ultrasonic sensor to play a song when it is less than 4cm from an object and stop playing the song when it is more than 25cm from an object.
* Include the list of the additional components you used
  + 1 Piezo Buzzer
  + 1 HC-SR04 ultrasonic sensor
* The hardware implementation
  + The positive end of the buzzer is plug into port A0
  + The trigger is plug into D3 and the echo is plugged into D4
  + Both the buzzer and ultrasonic sensor are plugged into ground and the ultrasonic sensor is plugged into the power
* The software implementation
  + In order to implement the software of each dance move, it would seem like there were two thread running, however, there was only one. To be checking the outer environment constantly, we would switch between moving the robot, checking the ultrasonic sensor and playing the buzzer. To play the buzzer, we have two lists; one for the note and another one for how long the note plays. Every time a note is played, the index was incremented. At the beginning of each dance move, the song is checked to determine if the song has finished and the song restarts (where the index goes to zero). For the ultrasonic sensor, we would check if the distance was less than 4cm to determine if it plays a song. While using the buzzer and ultrasonic sensor, the robot would continue to dance. Once one cycle was done, the software would re-check the components, and continue to do this in a loop.

**D. Test and evaluations**

Explain your evaluation and testing procedures for hardware and software. Please demonstrate systematic testing, debugging and continuous integration. Include the problems you have encountered and how you resolve them, as well as best practices you have incorporated.

While doing this project, our goal was to have a bug free robot. This demonstrates the need for testing and shows its importance to our overall product. To ensure that our integration easily allows for debugging, every individual component was thoroughly tested before it was added into our main functionality. For example, when preparing the robot, we tested each servo motor with the previously written raspberry pi code before implementing putting the new servo onto the robot. By doing this, we found out that a motor was not working and was able to replace it before testing our newly developed code.

With respect to testing with the code, each partition was coded separately and tested rigorously. This made it simple and easy to find errors in the code and allowed for solutions to be implemented. From there, we would integrate that part of the code into the main robot function. This process reduced the number of bugs in our code and allowed us to speed up the integration stages. When we did find bugs, we were able to remove it from integration and test it separately again before reintroducing it into our project.

When analyzing initial concerns with the robot, the first one that popped to mind was how the motors were receiving power. Only three motors can run at a time. At first, we attempted to reduce the noise by adding capacitors, but this failed to work. To fix this problem, we looked more closely at the documentation of the motors and the ItsyBitsy. Once doing this, we realized that you could provide power using the USB power source, not the 3V power source, and this resolved all of our concerns.

**E. Conclusions and Reflections**

Reflect and conclude on the lessons, tricks or interesting concepts you have learned during the project.

Also reflect on other aspects such as team work, project management, time management, ...

As this was our first project working as a full group, we started off by finding our respective roles that made us most comfortable and testing out the new group dynamic. Roles of leaders, mediators, and others came quite naturally to us and we were very quickly able to act as a fully functioning team. We easily divided different tasks amongst ourselves and split off during lab hours to work on our tasks. The management did not fall on just one person’s shoulders, but rather the task delegation was taken up by almost everyone. As for time management, we had a bit of trouble with this in the beginning as we all have the same upcoming deadlines and exam schedule. But we were able to still find days between exams to meet after class to work on our project.

As for what we learned regarding the project itself, this was the first time for some of our group members to be able to work on code that produced tangible movement. As a lot of experience we’ve had up until now has purely been displayed on computer screens, this was a great introduction project to see the connections between software and physical outcomes. We learned how we could produce intricate and exciting dance moves by building upon our simple functionalities. By simply just adjusting angles on motors, we were able to create patterns and repetitions in movement and create something both entertaining and fluid to watch. Overall, the open-endedness of the project also allowed for a lot of creative designing and processes. We were able to experiment with buzzers and keypads and choose which features we wanted to include in our end product. We’ve learned about many different designing strategies throughout first and second year and, from this project, we were able to put this knowledge to use and actually experience working with an iterative design process.

**F. References and bibliography**

Provide any relevant references.

Also include the list and description of the files submitted for this lab (including code and Fritzing breadboard schematics)

On our github, as individual members attempted writing dance moves themselves, our dances are all in separate files. We also created the buzzer and sonar stopping program separately. In submission, we have combined all of the dance moves and additional features into one main program.

* Main with dance moves, buzzer, and sonar
* Fritzing (breadboard view)
* Lab report (this file)

Include the following appendixes. There is no page limit for the appendixes.

**Appendix A – Robot pictures**

**A circuit board

Description automatically generatedA circuit board

Description automatically generatedA close up of electronics

Description automatically generatedA desk with a computer

Description automatically generated**Include a few pictures of your robot here. The pictures should clearly show the robot as a whole, as well as all electronics, wiring and parts. Include photos taken from the top, and from the sides. Show the location/installation of circuits and components as clearly as possible.

**A circuit board

Description automatically generatedA circuit board

Description automatically generatedA circuit board

Description automatically generated**

**Appendix B – Code**

All the code is clearly commented. In github, you can see that all the different parts of the robot were initially tested in their own files. This is was done whether it was a dance, buzzer or the integration between multiple components.

Include the complete CircuitPython code with comment statements. This code must be the same code as the files you demo and submit. Clearly identify the portion of the code for the main functionality and the Additional functionality.

The code must be readable, with indentation, syntax highlighting (that is, copy with colour coding), and on white background. The code must be in text (that is, absolutely no snapshots of the code).

*# adding import statements*

import board

import displayio

import terminalio

from adafruit\_display\_text import label

from adafruit\_st7735r import ST7735R

import adafruit\_imageload

import time

import pulseio

from adafruit\_motor import servo

import simpleio

import adafruit\_hcsr04

*# create a PWMOut object on Pin A2.*

*# can change inputs board.A1 etc.*

pwm1 = pulseio.PWMOut(board.D13, duty\_cycle=2 \*\* 15, frequency=50)

pwm2 = pulseio.PWMOut(board.D12, duty\_cycle=2 \*\* 15, frequency=50)

pwm3 = pulseio.PWMOut(board.D10, duty\_cycle=2 \*\* 15, frequency=50)

pwm4 = pulseio.PWMOut(board.D5, duty\_cycle=2 \*\* 15, frequency=50)

*# Create a servo object, my\_servo.*

lowLeft = servo.Servo(pwm1) *#alternative name servo1*

lowRight = servo.Servo(pwm2) *#alternative name servo2*

upRight = servo.Servo(pwm3) *#alternative name servo3*

upLeft = servo.Servo(pwm4) *#alternative name servo4*

*# setting up sonar to use pin out D3 and D4*

sonar = adafruit\_hcsr04.HCSR04(trigger\_pin=board.D3, echo\_pin=board.D4)

*# Release any resources currently in use for the displays*

displayio.release\_displays()

*# setting up LCD board*

spi = board.SPI()

tft\_cs = board.D11

tft\_dc = board.D9

*# setting up the LCD board so the code knows the size of the screen*

display\_bus = displayio.FourWire(spi, command=tft\_dc, chip\_select=tft\_cs, reset=board.D7)

display = ST7735R(display\_bus, width=128, height=128)

*# Define pin connected to piezo buzzer.*

*# creating a bunch of note varriables to help create the song*

PIEZO\_PIN = board.A1

NOTE\_B0  =31

NOTE\_C1  =33

NOTE\_CS1 =35

NOTE\_D1  =37

NOTE\_DS1 =39

NOTE\_E1  =41

NOTE\_F1  =44

NOTE\_FS1 =46

NOTE\_G1  =49

NOTE\_GS1 =52

NOTE\_A1  =55

NOTE\_AS1 =58

NOTE\_B1  =62

NOTE\_C2  =65

NOTE\_CS2 =69

NOTE\_D2  =73

NOTE\_DS2 =78

NOTE\_E2  =82

NOTE\_F2  =87

NOTE\_FS2 =93

NOTE\_G2  =98

NOTE\_GS2 =104

NOTE\_A2  =110

NOTE\_AS2 =117

NOTE\_B2  =123

NOTE\_C3  =131

NOTE\_CS3 =139

NOTE\_D3  =147

NOTE\_DS3 =156

NOTE\_E3  =165

NOTE\_F3  =175

NOTE\_FS3 =185

NOTE\_G3  =196

NOTE\_GS3 =208

NOTE\_A3  =220

NOTE\_AS3 =233

NOTE\_B3  =247

NOTE\_C4  =262

NOTE\_CS4 =277

NOTE\_D4  =294

NOTE\_DS4 =311

NOTE\_E4  =330

NOTE\_F4  =349

NOTE\_FS4 =370

NOTE\_G4  =392

NOTE\_GS4 =415

NOTE\_A4  =440

NOTE\_AS4 =466

NOTE\_B4  =494

NOTE\_C5  =523

NOTE\_CS5 =554

NOTE\_D5  =587

NOTE\_DS5 =622

NOTE\_E5  =659

NOTE\_F5  =698

NOTE\_FS5 =740

NOTE\_G5  =784

NOTE\_GS5 =831

NOTE\_A5  =880

NOTE\_AS5 =932

NOTE\_B5  =988

NOTE\_C6  =1047

NOTE\_CS6 =1109

NOTE\_D6  =1175

NOTE\_DS6 =1245

NOTE\_E6  =1319

NOTE\_F6  =1397

NOTE\_FS6 =1480

NOTE\_G6  =1568

NOTE\_GS6 =1661

NOTE\_A6  =1760

NOTE\_AS6 =1865

NOTE\_B6  =1976

NOTE\_C7  =2093

NOTE\_CS7 =2217

NOTE\_D7  =2349

NOTE\_DS7 =2489

NOTE\_E7  =2637

NOTE\_F7  =2794

NOTE\_FS7 =2960

NOTE\_G7  =3136

NOTE\_GS7 =3322

NOTE\_A7  =3520

NOTE\_AS7 =3729

NOTE\_B7  =3951

NOTE\_C8  =4186

NOTE\_CS8 =4435

NOTE\_D8  =4699

NOTE\_DS8 =4978

*# this allows us to set up how long each note plays for*

wholeNote = 2

halfNote = 1

quarterNote = 0.5

eighthNote = 0.25

sixteenthNote = 0.125

*# creating a list of notes*

notes = [NOTE\_C4, NOTE\_C4, NOTE\_D4, NOTE\_C4, NOTE\_F4, NOTE\_E4]

*# this is a list for our song shootingStars this list of notes corresponds with the list of notes in the shooting Stars Beats*

shootingStarsNotes = [

    NOTE\_DS5, NOTE\_DS5, 0, NOTE\_E5, NOTE\_B4, NOTE\_GS4, NOTE\_DS5,

    NOTE\_DS5, NOTE\_DS5, 0, NOTE\_E5, NOTE\_B4, NOTE\_GS4, NOTE\_DS5,

    NOTE\_DS5, NOTE\_DS5, 0, NOTE\_E5, NOTE\_B4, NOTE\_GS4, NOTE\_DS5,

    NOTE\_DS5, NOTE\_DS5, 0, NOTE\_E5, NOTE\_DS5, NOTE\_B4, NOTE\_GS4,

    NOTE\_FS5, NOTE\_FS5, NOTE\_GS5, NOTE\_DS5, NOTE\_B4, NOTE\_FS5,

    NOTE\_FS5, NOTE\_FS5, NOTE\_GS5, NOTE\_DS5, NOTE\_B4, NOTE\_FS5,

    NOTE\_FS5, NOTE\_FS5, NOTE\_GS5, NOTE\_DS5, NOTE\_B4, NOTE\_FS5,

    NOTE\_FS5, NOTE\_FS5, NOTE\_DS5, NOTE\_E5, NOTE\_DS5, NOTE\_B4, NOTE\_GS4,

    NOTE\_DS5, NOTE\_DS5, 0, NOTE\_E5, NOTE\_B4, NOTE\_GS4, NOTE\_DS5,

    NOTE\_DS5, NOTE\_DS5, 0, NOTE\_E5, NOTE\_B4, NOTE\_GS4, NOTE\_DS5,

    NOTE\_DS5, NOTE\_DS5, 0, NOTE\_E5, NOTE\_B4, NOTE\_GS4, NOTE\_DS5,

    NOTE\_DS5, NOTE\_DS5, 0, NOTE\_E5, NOTE\_DS5, NOTE\_B4, NOTE\_GS4

    ]

*# this is a list that has the length of each note above*

shootingStarsBeats = [

    quarterNote+eighthNote, sixteenthNote, sixteenthNote, quarterNote, eighthNote, sixteenthNote, sixteenthNote,

    quarterNote+eighthNote, sixteenthNote, sixteenthNote, quarterNote, eighthNote, sixteenthNote, sixteenthNote,

    quarterNote+eighthNote, sixteenthNote, sixteenthNote, quarterNote, eighthNote, sixteenthNote, sixteenthNote,

    quarterNote+eighthNote, sixteenthNote, sixteenthNote, eighthNote, eighthNote, eighthNote, eighthNote,

    quarterNote+eighthNote, eighthNote, quarterNote, eighthNote, sixteenthNote, sixteenthNote,

    quarterNote+eighthNote, eighthNote, quarterNote, eighthNote, sixteenthNote, sixteenthNote,

    quarterNote+eighthNote, eighthNote, quarterNote, eighthNote, sixteenthNote, sixteenthNote,

    quarterNote+eighthNote, eighthNote, eighthNote, eighthNote, eighthNote, sixteenthNote, sixteenthNote,

    quarterNote+eighthNote, sixteenthNote, sixteenthNote, quarterNote, eighthNote, sixteenthNote, sixteenthNote,

    quarterNote+eighthNote, sixteenthNote, sixteenthNote, quarterNote, eighthNote, sixteenthNote, sixteenthNote,

    quarterNote+eighthNote, sixteenthNote, sixteenthNote, quarterNote, eighthNote, sixteenthNote, sixteenthNote,

    quarterNote+eighthNote, sixteenthNote, sixteenthNote, eighthNote, eighthNote, eighthNote, eighthNote

    ]

*# a list of notes for mii*

miiChannelNotes = [

    NOTE\_FS5, NOTE\_A5, NOTE\_CS6, NOTE\_A5, NOTE\_FS5,

    NOTE\_D5, NOTE\_D5, NOTE\_D5, 0, NOTE\_CS5,

    NOTE\_D5, NOTE\_FS5, NOTE\_A5, NOTE\_CS6, NOTE\_A5, NOTE\_FS5,

    NOTE\_E6, NOTE\_DS6, NOTE\_D6, 0,

    NOTE\_GS5, 0, NOTE\_CS6, NOTE\_FS5, 0, NOTE\_CS6, 0, NOTE\_GS5,

    0, NOTE\_CS6, 0, NOTE\_A5, NOTE\_FS5, 0, NOTE\_E5, 0,

    NOTE\_E5, NOTE\_E5, NOTE\_E5, 0, NOTE\_E5, NOTE\_E5,

    NOTE\_E5, 0, NOTE\_DS5, NOTE\_D5, NOTE\_CS5

]

*# a list of beat length. Each time has a note that it matches in the list "miiChannelNotes"*

miiChannelBeats = [

    quarterNote, eighthNote, quarterNote, quarterNote, eighthNote,

    eighthNote, eighthNote, eighthNote, halfNote, eighthNote,

    eighthNote, eighthNote, eighthNote, quarterNote, quarterNote, eighthNote,

    quarterNote, eighthNote, eighthNote, quarterNote+eighthNote,

    eighthNote, eighthNote, eighthNote, eighthNote, eighthNote, eighthNote, eighthNote, eighthNote,

    eighthNote, eighthNote, eighthNote, eighthNote, eighthNote, eighthNote, eighthNote, eighthNote,

    eighthNote, eighthNote, eighthNote, quarterNote+eighthNote, eighthNote, eighthNote,

    eighthNote, eighthNote+quarterNote, quarterNote, quarterNote

]

*# Display image on the LCD*

**def** displayImg(fileName):

*# Loads the bitmap image*

    bitmap, palette = adafruit\_imageload.load(fileName, bitmap=displayio.Bitmap, palette=displayio.Palette)

*# Create a TileGrid to store the bitmap*

    tile\_grid = displayio.TileGrid(bitmap, pixel\_shader=palette)

*# Create a Group to store the TileGrid*

    group = displayio.Group()

*# Add the titleGrid to the Group*

    group.append(tile\_grid)

*# Show the group*

    display.show(group)

**def** miiProgram():

    lowLeft.angle = 90 *# set left foot angle*

    lowRight.angle = 90 *# set right foot angle*

    upLeft.angle = 90 *# set left knee angle*

    upRight.angle = 90 *# set right knee angle*

*# this makes the robot face forward*

    while 1:

        for index in range(0, len(miiChannelBeats), 1):

            simpleio.tone(PIEZO\_PIN, miiChannelNotes[index], duration=(miiChannelBeats[index])/1.5) *# play the miiChannel Song*

            try:

                if (sonar.distance > 25):

                    return 0 *# if the distance of the sonar is greater than 25cm stop playing the song*

            except RuntimeError:

                print("Retrying!")

            time.sleep(0.1) *# wait 0.1 seconds*

*# Check if something is within 4cm of sonar*

**def** checkSonar():

    try:

        distance = sonar.distance *# get the distance from the sonar*

*# check if distance is less than 4cm*

        if (distance < 4):

            miiProgram() *# if the sonor distance is less than 4cm run the miiProgram*

    except RuntimeError:

        print("Retrying!") *# error message*

index = 0   *# Initialize global variable to 0*

**def** dance1():

**global** index *# make sure the global varriable index can be accessed*

    index = 0 *# this is the very first dance move. Set index to zero so restart the shooting star song*

    displayImg("/fireFeet.bmp") *# Display corresponding img on LCD*

    print("Dance 1") *# print the word Dance 1*

    lowLeft.angle = 90 *# set angle of left foot*

    lowRight.angle = 90 *# set angle of right foot*

    upLeft.angle = 90 *# set angle of left knee*

    upRight.angle = 90 *# set angle of right foot*

    for num in range(5):

        lowLeft.angle = 60 *# set left foot*

*# Because we only have one processor and we want to do two things at the same time we make it seem like it is threading*

*# so we play the buzzer*

*# the buzzer is given a note from the list shootingStarsNotes and its corresponding duration from shootingStarsBeats*

*# both of these places are determined by the global variable index which is increased by 1 after each note*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        lowLeft.angle = 120 *# set left foot angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        lowLeft.angle = 60 *# set left foot angle*

    lowLeft.angle = 90 *# set left foot angle*

    for num in range(5):

        upLeft.angle = 60 *# set left knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        upLeft.angle = 120 *# set left knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        upLeft.angle = 60 *# set left knee angle*

    for num in range(5):

        lowRight.angle = 120 *# set right foot angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        lowRight.angle = 60 *# set right foot angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        lowRight.angle = 120 *# set right foot angle*

    lowRight.angle = 90 *# set right foot angle*

    for num in range(5):

        upRight.angle = 120 *# set right knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        upRight.angle = 60 *# set right knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        upRight.angle = 120 *# set right knee angle*

**def** dance2():

    print("Dance 2")

    displayImg("/jumpingJacks.bmp")  *#display corresponding image on LCD*

**global** index *# make sure the global varriable index can be accessed*

    if (index >= len(shootingStarsBeats) - 30):

        index = 0 *# if index is greater than the shootingStar song restart the song*

    lowLeft.angle = 90 *# set left foot angle*

    lowRight.angle = 90 *# set right foot angle*

    upLeft.angle = 90 *# set left knee angle*

    upRight.angle = 90 *# set right knee angle*

    time.sleep(0.1) *# sleep for 1 second*

    count = 0

    while count < 2:

*# run this loop twice*

        upLeft.angle = 130 *# set left knee anlge*

        upRight.angle = 50 *# set right knee angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        upLeft.angle = 50 *# set left knee angle*

        upRight.angle = 130 *# set right knee angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        lowLeft.angle = 110 *# set left foot angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        lowLeft.angle = 90 *# set left foot angle*

        lowRight.angle = 60 *# set right foot angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        lowRight.angle = 90 *# set right foot angle*

        lowLeft.angle = 110 *# set left foot angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        lowLeft.angle = 90 *# set left foot angle*

        lowRight.angle = 60 *# set right foot angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        lowRight.angle = 90 *# set right foot angle*

        lowLeft.angle = 110 *# set left foot angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        lowLeft.angle = 90 *# set left foot angle*

        lowRight.angle = 60 *# set right foot angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        lowRight.angle = 90 *# set right foot angle*

        count += 1 *# increases count by 1*

**def** dance3():

    print("Dance 3")

    displayImg("/shuffle.bmp")  *#display corresponding image on LCD*

**global** index *# make sure the global varriable index can be accessed*

    if (index >= len(shootingStarsBeats) - 30):

        index = 0 *# if index is greater than the shootingStar song restart the song*

    lowLeft.angle = 90 *# set left foot angle*

    lowRight.angle = 90 *# set right foot angle*

    upLeft.angle = 90 *# set left knee angle*

    upRight.angle = 90 *# set right knee angle*

    time.sleep(0.1) *# sleep for 0.1 seconds*

    for num in range(60, 120, 20):

        if num % 40 == 0:

*# if num is a multiple of 40 do the following*

            lowLeft.angle = num + 10 *# set left foot angle*

            lowRight.angle = num - 10 *# set right foot angle*

*# Because we only have one processor and we want to do two things at the same time we make it seem like it is threading*

*# so we play the buzzer*

*# the buzzer is given a note from the list shootingStarsNotes and its corresponding duration from shootingStarsBeats*

*# both of these places are determined by the global varrialbe index which is increased by 1 after each note*

            simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

            index = index + 1 *# increase the index by 1*

            checkSonar()

        lowLeft.angle = num *# set left foot angle*

        lowRight.angle = 180 - num *# set right foot angle*

        upLeft.angle = num *# set left knee angle*

        upRight.angle = 180 - num *# set right knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

    for num in range(120, 60, -20):

        upRight.angle = num *# set right knee angle*

        upLeft.angle = 180 - num *# set left knee angle*

        upLeft.angle = num *# set left knee angle*

        upRight.angle = 180 - num *# set right knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        if num % 40 == 0:

*# if num is a multiple of 40 do the following*

            upRight.angle = num - 10 *# set right knee angle*

            upLeft.angle = num + 10 *# set left knee angle*

            simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

            index = index + 1 *# increase the index by 1*

            checkSonar()

**def** dance4():

    print("Dance 4")

    displayImg("/step.bmp") *# Display corresponding img on LCD*

**global** index *# make sure the global varriable index can be accessed*

    if (index >= len(shootingStarsBeats) - 30):

        index = 0 *# if index is greater than the shootingStar song restart the song*

    for angle in range(70, 110, 20):  *# 30 - 150 degrees, 5 degrees at a time.*

        lowLeft.angle = angle *# set left foot angle*

*# Because we only have one processor and we want to do two things at the same time we make it seem like it is threading*

*# so we play the buzzer*

*# the buzzer is given a note from the list shootingStarsNotes and its corresponding duration from shootingStarsBeats*

*# both of these places are determined by the global varrialbe index which is increased by 1 after each note*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        lowRight.angle = 180 - angle *# set right foot angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        upRight.angle = angle *# set right knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        upLeft.angle = 180 - angle *# set left knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

    for angle in range(110, 70, -20):  *# 150 - 30 degrees, 5 degrees at a time.*

        lowLeft.angle = angle *# set left foot angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        lowRight.angle = 180 - angle *# set right foot angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        upRight.angle = angle *# set right knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        upLeft.angle = 180 - angle *# set left knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

    for angle in range(70, 110, 20):  *# 150 - 30 degrees, 5 degrees at a time.*

        if angle % 90 == 0:

*# if the angle is a multiple of 90 do the following*

            lowLeft.angle = angle - 10 *# set left foot angle*

            simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

            index = index + 1 *# increase the index by 1*

            checkSonar()

            lowRight.angle = 180 - angle + 10 *# set right foot angle*

            simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

            index = index + 1 *# increase the index by 1*

            upRight.angle = angle - 10 *# set right knee angle*

            simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

            index = index + 1 *# increase the index by 1*

            checkSonar()

            upLeft.angle = 180 - angle + 10 *# set left knee angle*

            simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

            index = index + 1 *# increase the index by 1*

        lowLeft.angle = angle *# set left foot angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        lowRight.angle = 180 - angle *# set right foot angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        upRight.angle = angle *# set right knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        upLeft.angle = 180 - angle *# set left knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index])  *# play the buzzer*

        index = index + 1 *# increase the index by 1*

    for angle in range(110, 70, -20):  *# 150 - 30 degrees, 5 degrees at a time.*

        if angle % 90 == 0:

*# if angle is a multiple of 90 do the following*

            lowLeft.angle = angle - 20 *# set left foot angle*

            time.sleep(0.1) *# sleep for 0.1 seconds*

            checkSonar()

            lowRight.angle = 180 - angle + 20 *# set right foot angle*

            time.sleep(0.1) *# sleep for 0.1 seconds*

            upRight.angle = angle - 20 *# set right knee angle*

            time.sleep(0.1) *# sleep for 0.1 seconds*

            checkSonar()

            upLeft.angle = 180 - angle + 20 *# set left knee angle*

            time.sleep(0.1) *# sleep for 0.1 seconds*

    lowLeft.angle = angle *# set left foot angle*

    simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

    index = index + 1 *# increase the index by 1*

    checkSonar()

    lowRight.angle = 180 - angle *# set right foot angle*

    simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

    index = index + 1 *# increase the index by 1*

    upRight.angle = angle *# set right knee angle*

    simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

    index = index + 1 *# increase the index by 1*

    checkSonar()

    upLeft.angle = 180 - angle *# set left knee angle*

    simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

    index = index + 1 *# increase the index by 1*

**def** dance5():

    print("Dance 5")

    displayImg("/dancing.bmp") *# Display corresponding img on LCD*

**global** index *# make sure the global varriable index can be accessed*

    if (index >= len(shootingStarsBeats) - 30):

        index = 0 *# if index is greater than the shootingStar song restart the song*

    count = 0

*# this dance makes it shift backwards*

    while count < 5:

        upRight.angle = 60 *# set right knee angle*

        upLeft.angle = 60 *# set left knee angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        upLeft.angle = 120 *# set left knee angle*

        upRight.angle = 120 *# set right knee angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        upRight.angle = 60 *# set right knee angle*

        upLeft.angle = 60 *# set left knee angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        upRight.angle = 120 *# set right knee angle*

        upLeft.angle = 120 *# set left knee angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        upRight.angle = 60 *# set right knee angle*

        upLeft.angle = 60 *# set left knee angle*

        checkSonar()

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        count += 1 *# increase count by 1*

**def** dance6():

    print("Dance 6")

    displayImg("/TipToe.bmp") *# Display corresponding img on LCD*

**global** index *# make sure the global varriable index can be accessed*

    if (index >= len(shootingStarsBeats) - 30):

        index = 0 *# if index is greater than the shootingStar song restart the song*

    count = 0 *# set count to 0*

    while count < 3:

*# run this set of code 3 times*

        lowLeft.angle = 90 *# set left foot angle*

        lowRight.angle = 90 *# set right foot angle*

        upLeft.angle = 90 *# set left knee angle*

        upRight.angle = 90 *# set right knee angle*

        for angle in range(60, 120, 10):  *# 0 - 180 degrees, 5 degrees at a time.*

            upLeft.angle = angle *# set left knee angle*

            upRight.angle = 180 - angle *# set right knee angle*

*# Because we only have one processor and we want to do two things at the same time we make it seem like it is threading*

*# so we play the buzzer*

*# the buzzer is given a note from the list shootingStarsNotes and its corresponding duration from shootingStarsBeats*

*# both of these places are determined by the global varrialbe index which is increased by 1 after each note*

            simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

            index = index + 1 *# increase the index by 1*

            checkSonar()

            if (angle == 90):

                lowRight.angle = 120 *# set right foot angle*

                lowLeft.angle = 60  *# set left foot angle*

                simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

                index = index + 1 *# increase the index by 1*

                lowRight.angle = 60 *# set right foot angel*

                lowLeft.angle = 120 *# set left foot angle*

                simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

                index = index + 1 *# increase the index by 1*

                checkSonar()

                lowRight.angle = 120 *# set right foot angle*

                lowLeft.angle = 60 *# set left foot angle*

                simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

                index = index + 1 *# increase the index by 1*

        for angle in range(60, 120, -10):  *# 0 - 180 degrees, 5 degrees at a time.*

            upLeft.angle = angle *# set left knee angle*

            upRight.angle = angle *# set right knee angle*

            simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

            index = index + 1 *# increase the index by 1*

            checkSonar()

            if (angle == 90):

*# run this code only if the angle is 90 degrees*

                lowRight.angle = 120 *# set right foot angle*

                lowLeft.angle = 60 *#set left foot angle*

                simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

                index = index + 1 *# increase the index by 1*

                lowRight.angle = 60 *# set right foot angle*

                lowLeft.angle = 120 *# set left foot angle*

                simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

                index = index + 1 *# increase the index by 1*

                checkSonar()

                lowRight.angle = 120 *# set right foot angle*

                lowLeft.angle = 60 *# set left foot angle*

                simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

                index = index + 1 *# increase the index by 1*

        upLeft.angle = 90 *# set left knee angle*

        upRight.angle = 90 *# set right knee angle*

        simpleio.tone(PIEZO\_PIN, shootingStarsNotes[index], duration=shootingStarsBeats[index]) *# play the buzzer*

        index = index + 1 *# increase the index by 1*

        checkSonar()

        count+=1 *# increase the count by 1*

lowLeft.angle = 90 *# set left foot angle*

lowRight.angle = 90 *# set right foot angle*

upLeft.angle = 90 *# set left knee angle*

upRight.angle = 90 *# set right knee angle*

time.sleep(1) *# sleep for one second*

while True:

    pass

*# initalDisplay()*

    displayImg("/intro.bmp")

    time.sleep(2)

*# this is our main function. It runs on a loop and has all our dances*

    dance1() *# dance 1*

    dance2() *# dance 2*

    dance3() *# dance 3*

    dance4() *# dance 4*

    dance5() *# dance 5*

    dance6() *# dance 6*

    time.sleep(0.5)

    displayImg("/splits.bmp")

*# do the splits*

    upLeft.angle = 90 *# set left knee angle*

    upRight.angle = 90 *# set right knee angle*

    lowLeft.angle = 160 *# set left foot angle*

    lowRight.angle = 20 *# set right foot angle*

    time.sleep(5) *# wait 5 seconds to restart the dance*

**Appendix C – Fritzing**

**A picture containing screenshot

Description automatically generated**Include the snapshot of your fritzing breadboard-view schematic. Include as many as you have, but clearly describe which is which. This is in addition to the fritzing file that you submit to the Canvas.

Below is a photo of the fritzing. It has been submitted on canvas and is also found on github.

**Appendix D – GitHub**

Every group member must have reasonably and equally contributed to the project github repository. If that is not the case for any member and there is a valid reason as to why, please include an explanation here.

**Appendix E – Complete Component list**

Include the list of all the components used for the project.

If you have used any component you have purchased on own, include full info, a link to datasheet, and cost.

* One Robot hull
* 2 robot feet
* 4 servo motors
* HC-SR04 Ultrasonic Sensor
* 1 Adafruit ItsyBitsy M4
* 1 Adafruit 1.44 inch TFT Breakout
* 1 Piezo Speaker
* A whole lot of wire

**Appendix F – Answer the following questions:**

Q1 – Teamwork: Explain in details the methods your group has used to communicate effectively among team members.

To communicate with one another, we created a Facebook messenger group chat. This allowed us to communicate effectively and quickly, especially during our reading break. In addition, we developed a Gantt chart to distribute the work and ensure we remained on schedule. This chart kept track of tasks that needed to be done, who oversaw the task and when it was due.

Q2 – Design Process for the additional functionalities: Describe clearly the process you used for the following design aspects of your own additional functialities. Please spend time to carefully answer each of them.

1. **Use of process**: Describe your approach to adapt and apply a general design process for any additional feature. What was your approach?

When conducting this project, we followed the Gantt chart and divided the work equally. For more difficult aspects of the project, two people or all of us would meet to work together to develop solutions and build what was necessary. This included building the robot, programing the LCD screen and working on the buzzer or the ultrasonic sensor. After doing our individual parts, we would review the one another’s code. This helped us quickly fix bugs, get a new perspective on our individual task and integrate these components quickly into the final product.

1. **Constraint identification**: Explain the constraints that you must consider in the design of the additional functionalities.
   1. **Constraint 1**: The biggest constraint we had was the amount of ports on the ItsyBitsy. To fix this constraint, we could have used a shift register.
   2. **Constraint 2**: When introducing additional features, we found that it would sometimes slow down the main functionality of our program. For example, when syncing the moves with the music, we attempted to introduce our own version of threading by continuously switching between a movement and then a buzzer sound. We found that when we executed this, it made the dance moves look very rigid and broken. We had to find a way to sync the music and moves while keeping the flow and movement appealing.
2. **Solution generation**: Explain at least two possible alternative additional features that your group rejected due to technical reasons and explain why.
   1. Because of the amount of ports on the ItsyBitsy, we decided to not go with the keypad. We could have integrated this into the robot with a shift register, however, but we decided that the benefit when compared to the amount of work required was not worth it.
   2. We also decided to not go with the LED light show. The ultrasonic sensor and buzzer would be better additions in place of the LED light show because they had more of an impact on the design and outcome of the robot, and allowed us more flexibility in our design.
3. **Solution Assessment**: Explain how you tested and assessed the viability and then correctness of your group’s additional features.
   1. To be successful in this project, testing, assessing viability and overall correctness was vital. Before integrating any part into the project, we tested the code and its implication extensively. This is shown in our GitHub repository. Each dance has its own file, and each one was tested rigorously. This reduced errors in the overall project and code and allowed our results to be more correct. This ideology was applied in all aspects of creating the robot.

**Appendix G - Other**

Include any other relevant info that does not fit in any other section in the report.