EE – 351 (Control & Robotics laboratory)

Module – 2 (A simple 2 player game design)

Vidhi Sahai

190102085

Abstract

This experiment comprises of 3 modules:

2a - Implement a 2-player game in Arduino (with one of the players being the Arduino). The rules of the game are:

1. The game is between the Arduino board (say Player A) and you (say Player B).

2. At first Player A will display a random number in the seven segment display (0-9). Say the number displayed is x.

3. Player B will have to input the number x mod 3 with the help of switches connected to the Arduino board.

4. To signal if the answer by Player B is correct or wrong, a buzzer would be used. If Player B is correct, then the buzzer would beep once. Else the buzzer would give three beeps to indicate that the answer is wrong. The three beeps for wrong answer should sound different.

5. Player B needs to input his/her answer within 10 seconds. The timer designedin Module 1 will be used to display the time. The timer should count from 0 - 9 and stop at the tenth second. This means that the timer should display 0 mat the end of the count.

Answer the following (Write brief and to-the-point answers):

1. Is the system designed by you a control system? (Give reasons for your answer)

2. What changes will you make to the module designed, if Player B has to input the number x mod 4?

3. What changes will you make to the module designed, if Player B has to input the number x mod 7?

4. What is the ON time of the square pulse that you have used for the piezobuzzer? What is its duty cycle?

5. Will it be possible to replace the Arduino with an actual player? If yes what changes will you need to do to the circuit and code that you have implemented?

2b - Design a game in TinkerCad using the circuit elements used in Module-02A (You cannot use the mod operation idea). Use the annotation option of TinkerCad to write down the rules of your game.

2c - Will it be possible to use the buzzer to design an automatic music player? For example, when the switch is ON, the buzzer should keep on playing the Bhopali Raag: Sa - Re - Ga - Pa - Dha - Sa’ followed by Sa’ - Dha - Pa - Ga - Re - Sa. If Yes, implement it. If No, justify. Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP (In-cicuit serial programming) header and a reset button. Embedded C is a set of language extensions for the C programming language by the C Standards Committee to address commonality issues that exist between C extensions for different embedded systems. Whenever the conventional C language and its extensions are used for programming embedded systems, it is referred to as “Embedded C” programming. We will be using the Arduino Uno R3 board along with a breadboard and some other basic electrical components to build all our circuit modules. We will be programming our Arduino Uno R3 using Embedded C programming language.

Module – 2a(Implement a 2-Player Game in Arduino)

In order to make this circuit, we need our Arduino Uno R3 to output the binary equivalent of the numbers 0 to 9. This can be done using just 4 bits. Hence, we assign the pins 10-13 as output pins (using pinMode(pin, OUTPUT) where pin goes from 10 to 13) inside void setup(){...} . We also need another similar circuit to display the randomly generated number. Hence, we assign the pins 6-9 as output pins (using pinMode(pin, OUTPUT) where pin goes from 6 to 9) inside void setup(){...} . We need a buzzer to play different sounds depending on whether the answer given by Player B is right or wrong. Hence, we assign pin 1 as the output pin for buzzer using #define BUZ 1 and pinMode(BUZ, OUTPUT) inside void setup(){...} . We need to get user input. For this, we use a DIP switch and keep pins 2-5 as the inputs coming from those switches. We define the four pins 5, 4, 3 and 2 as SW1, SW2, SW3 and SW4 respectively using #define statements. We assign these pins as input pins using pinMode(pin, INPUT) where pin goes from SW1 to SW4 inside void setup(){...} . Finally, we use randomSeed(analogRead(0)) so that we get a repeating long sequence of randomly generated numbers.

We make a 2-D array int NumberArray[11][4] , which corresponds to the binary equivalent of all numbers from 0 to 9 and 15. 15 is added so that at the end, the 7-segment display does not display anything. This array can be represented by the following table;

Number	Pins 6 & 10	Pins 7 & 11	Pins 8 & 12	Pins 9 & 13
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
NA	1	1	1	1

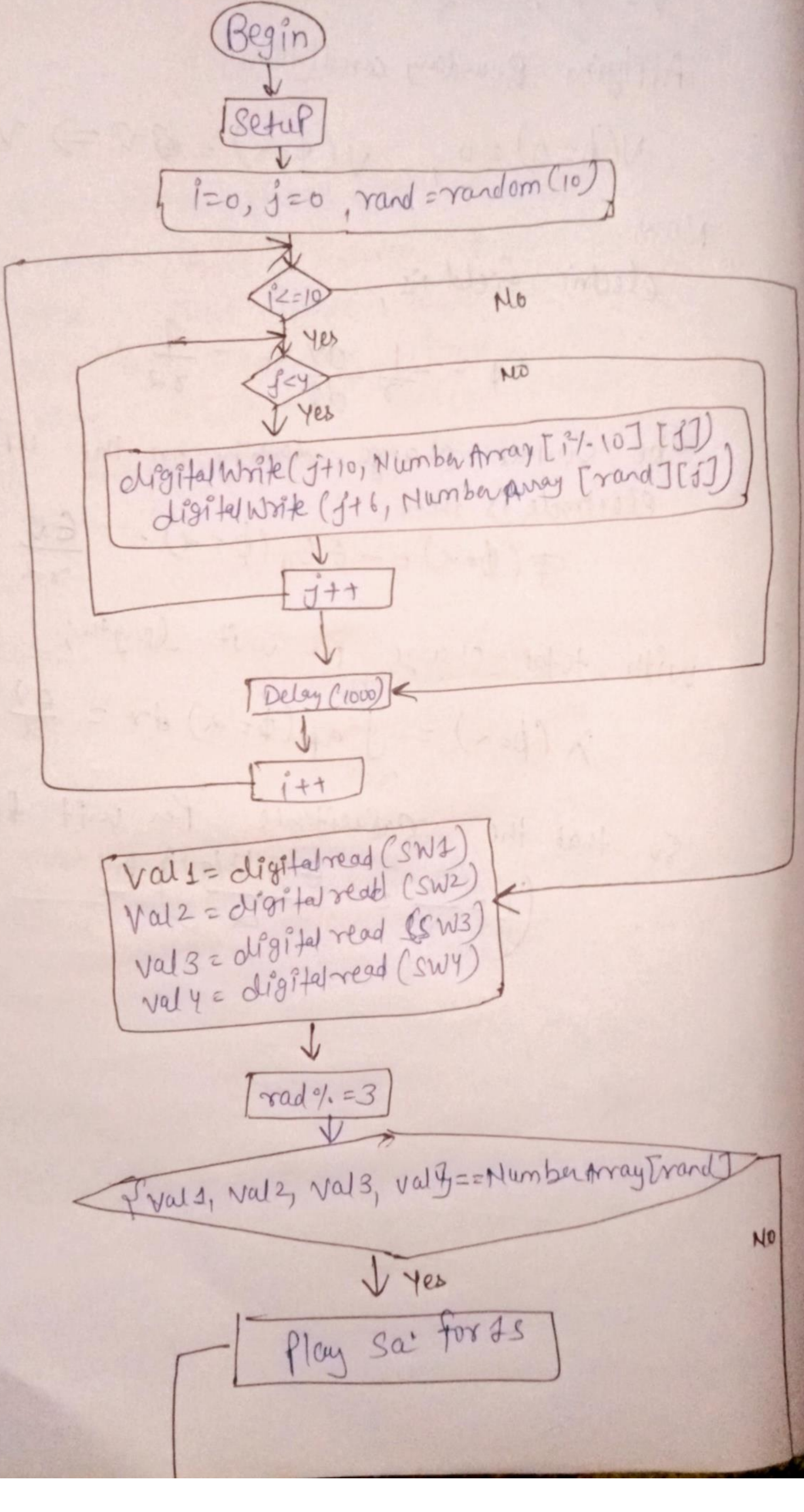
We declare four variables, int val1, val2, val3, val4 to hold the digital values read from the four switches. As we need to display the numbers 0 to 9, we declare two integers, int i = 0, j = 0 . We will use these to loop over the 2-D array NumberArray[11][4] . We also declare another integer, int rand = random(10); to hold the randomly generated number. Next, using nested for loops, we go over the numbers 0-9 and then assign to the output the bits 0-3 of each of these numbers using digitalWrite(j + 10, NumberArray[i % 10][j]) . As we also need to display the randomly generated number, we use digitalWrite(j + 6, NumberArray[rand][j]) alongside the previous statement. Now, after displaying every digit, we add a delay of 1s using delay(1000) . After all 10 numbers (0-9) have been displayed, the output becomes 0000, corresponding to 0 again, and the 7-segment display displays the digit '0'. Now, we read the Player B inputs into val1 , val2 , val3 and val4 . Also, as we need to check using the remainder that we get when dividing rand by 3, we use rand %= 3 .

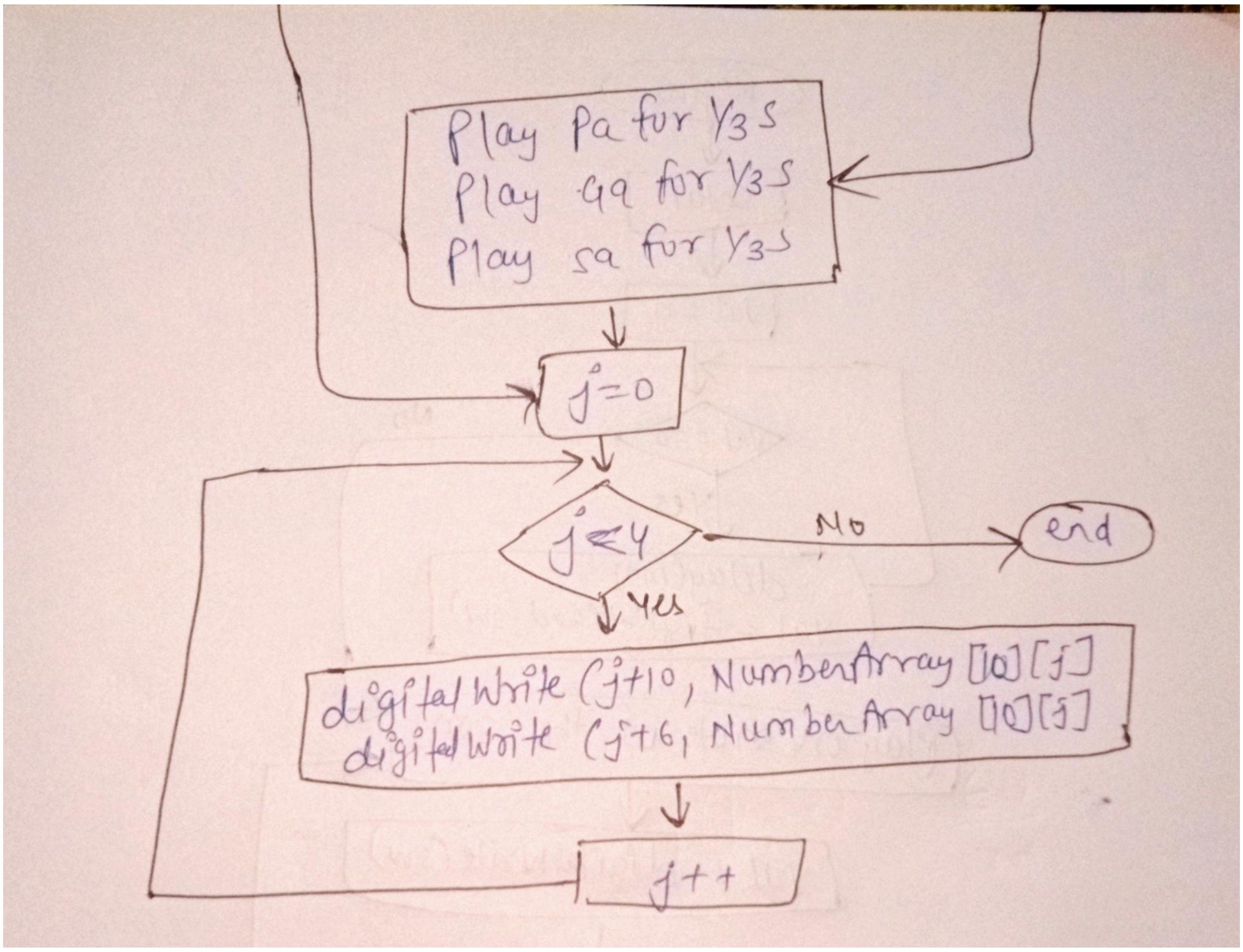
Now, we check if user input is right or wrong. Hence, we evaluate the truth value of the statement (val1 == NumberArray[rand][0]) && (val2 == NumberArray[rand][1]) && (val3 == NumberArray[rand][2]) && (val4 == NumberArray[rand][3]) . Now based on whether this is true or false, we generate different sounds from the buzzer. Hence, there are two cases:

1. Answer is right: In this case, we decide to generate a single sound for a duration of 1s. We choose to have this sound as the Sa' note (frequency = 523.251 Hz). Hence, we write tone(BUZ, 523.251, 1000) . Also, as we want this tone to be played for 1s, we write delay(1000).

2. Answer is wrong: In this case, we generate a 3-note descending sequence of frequencies, corresponding to the notes Pa (frequency = 391.995 Hz), Ga (frequency = 329.628 Hz) and Sa (frequency = 261.626 Hz). We want these notes to be played for 333 ms each. Hence, for playing Pa we write tone(BUZ, 391.995, 333) followed by delay(333) . We repeat these two statements twice for the remaining two frequencies in the sequence.

Flowchart


Begin
Setup
i=0, j=0 , rand = random (10)
i<=10
No
Yes
j<4
NO
Yes
digitalWrite (j+10, NumberArray [i%10] [j])
digitalWrite (j+6, NumberArray [rand][j])
j++
Delay (1000)
i++
Val 1 = digitalread (SW1)
Val 2 = digitalread (SW2)
Val 3 = digitalread (SW3)
val 4 = digitalread (SW4)
rad % =3
{val 1, val 2, val 3, val 4} == NumberArray [rand]
NO
Yes
Play Sa for 1s


Play Pa for 1/3 s
Play Ga for 1/3 s
Play sa for 1/3 s
j=0
j<4
No
end
Yes
digitalWrite (j+10, NumberArray [10][j]
digitalWrite (j+6, NumberArray [10][j]
j++

Arduino code

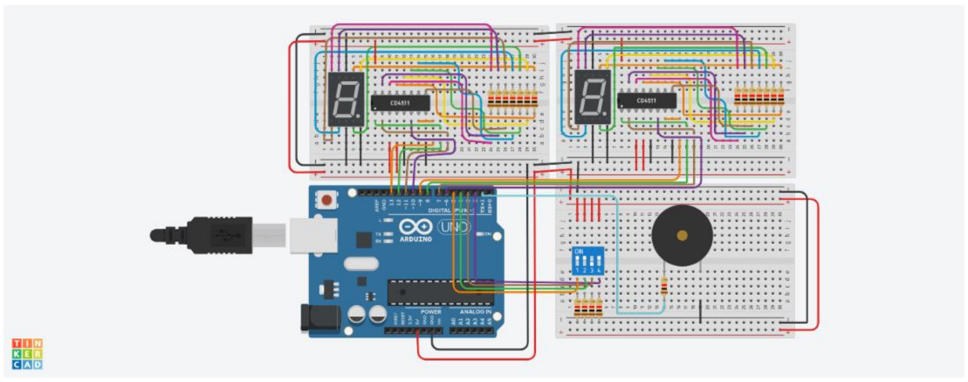
```
// C++ code

// define buzzer output as pin 1

#define BUZ 1

// define switch input as pins 2−5

#define SW4 2

#define SW3 3

#define SW2 4

#define SW1 5

// 2−D array representing 0 to 9 numbers

int NumberArray[11][4] = {{ 0 , 0 , 0 , 0 } , // 0

                          { 0 , 0 , 0 , 1 } , // 1

                          { 0, 0 , 1 , 0 } , // 2

                          { 0 , 0 , 1 , 1 } , // 3

                          { 0 , 1 , 0 , 0 } , // 4
```

```
                                    { 0 , 1 , 0 , 1 } , // 5
                                    { 0 , 1 , 1 , 0 } , // 6
                                    { 0 , 1 , 1 , 1 } , // 7
                                    { 1 , 0 , 0 , 0 } , // 8
                                    { 1, 0 , 0 , 1 } , // 9
                                    { 1, 1 , 1 , 1 } } ; // invalid input to stop displaying numbers on the 7−segment display
 int val1 , val2 , val3 , val4 ; // variables to hold switch input values
void setup ( )
{
// code for pins setup
// random number display output pins initialization
pinMode ( 6 , OUTPUT) ; // making pin 6 as 7−segment display output
pinMode ( 7 , OUTPUT) ; // making pin 7 as 7−segment display output
pinMode ( 8 , OUTPUT) ; // making pin 8 as 7−segment display output
pinMode ( 9 , OUTPUT) ; // making pin 9 as 7−segment display output
// timer count display output pins initialization
pinMode (10 , OUTPUT) ; // making pin 10 as 7−segment display output
pinMode (11 , OUTPUT) ; // making pin 11 as 7−segment display output
pinMode (12 , OUTPUT) ; // making pin 12 as 7−segment display output
pinMode (13 , OUTPUT) ; // making pin 13 as 7−segment display output
// switch intput pins initialization
pinMode (SW1, INPUT) ; // making pin 5 as first switch input
pinMode (SW2, INPUT) ; // making pin 4 as second switch input
pinMode (SW3, INPUT) ; // making pin 3 as third switch input
pinMode (SW4, INPUT) ; // making pin 2 as fourth switch input
pinMode (BUZ, OUTPUT) ; // making pin 6 as buzzer output
randomSeed ( analogRead ( 0 ) ) ; // random seed
// code for the main game
Int i = 0 , j = 0 ; // variables for l o oping over NumberArray
int rand = random (10) ; // variables to store random number
for ( i = 0 ; i <= 1 0 ; i++) // loop f o r numbers
{
for ( j = 0 ; j < 4 ; j++) // loop f o r b i t s o f each number
{
```

```
digitallWrite ( j + 10 , NumberArray [ i % 1 0 ] [ j ] ) ; // d i s pl a y count

digitalWrite( j+ 6 , NumberArray [ rand ] [ j ] ) ; // d i s p l a y random number

}

delay (1000) ; // wait 1 s be f o r e next assignment

 }

val1 = di g i t a lRe ad (SW1) ; // read f i r s t switch b i t

 val2 = di g i t a lRe ad (SW2) ; // read second switch b i t

val3 = di g i t a lRe ad (SW3) ; // read th i r d switch b i t

val4 = di g i t a lRe ad (SW4) ; // read f our th switch b i t

rand %= 3 ; // take modulo 3 o f the random number

// i f input equa l s the random number

if ( ( val1 == NumberArray [ rand ] [ 0 ] ) && ( val2 == NumberArray [ rand ] [ 1 ] ) && ( val3 == NumberArray [ rand ] [
2 ] ) && ( val4 == NumberArray [ rand ] [ 3 ] ) )

{

tone (BUZ, 523.251 , 1000) ; // play Sa '

delay (1000) ; // wait f o r Sa ' to have played

}

// i f input does not equal the random number

else

{

tone (BUZ, 391.995 , 333) ; // play Pa

delay (333) ; // wait f o r Pa to have played

tone (BUZ, 329.628 , 333) ; // play Ga

delay (333) ; // wait f o r Ga to have played

tone (BUZ, 261.626 , 333) ; // play Sa

delay (333) ; // wait f o r Sa to have played

}

// d i s p l a y nothing on both 7−segment d i s p l a y s

for ( j = 0 ; j < 4 ; j++) // loop f o r b i t s o f each number

{

digitalWrite ( j + 10 , NumberArray [ 1 0 ] [ j ] ) ; // d i s p l a y nothing

digitalWrite ( j + 6 , NumberArray [ 1 0 ] [ j ] ) ; // d i s p l a y nothing

}

}
```

```
void loop ( )

{

// nothing here as we do not want to repl ay the game

}
```

Circuit Diagram



Answers to Questions

1) Is the system designed by you a control system? (Give reasons for your answer)

Control is used to modify the behavior of a system so it behaves in a specific desirable way over time, without human intervention. Control is used whenever quantities such as speed, altitude, temperature, or voltage must be made to behave in some desirable way over time. Since our system controls digital voltages and will work even if Player B is idle (that is, there is no need of human intervention as such), we can clearly say that the system designed by us is indeed a control system.

2) What changes will you make to the module designed, if Player B has to input the number x mod 4?

For this, we just change the statement rand %= 3 to rand %= 4 at line 71 inside void setup{...} function of the code of module 2a.

3) What changes will you make to the module designed, if Player B has to input the number x mod 7?

For this, we just change the statement rand %= 3 to rand %= 7 at line 71 inside void setup{...} function of the code of module 2a.

4) What is the ON time of the square pulse that you have used for the piezo-buzzer? What is its duty cycle?

The square wave generated is of the specified frequency and 50% duty cycle. Hence, ON

time = 1/(2 × frequency). Hence, in case the answer given by player B is right, then ON time is 955.556 µs. In case the answer given by player B is wrong, then ON time changes as: 1.276 ms - 1.517 ms - 1.911 ms.

5) Will it be possible to replace the Arduino with an actual player? If yes what changes will you need to do to the circuit and code that you have implemented?

Yes, it is indeed possible. We will need to give player A a 4-bit DIP switch so that they can set the input number. The 7-segment display showing that input number must be removed, as otherwise we will run out of pins on the Arduino. We will give Player A 10 seconds to enter the number using the counter code we have already written. Next, instead of randomly generating the number, we will read it from the input pins corresponding to the DIP switch of Player A. In order to calculate whether the Player B is right or wrong, we can use the 2-D array NumberArray[11][4] to first convert the input bits to their decimal equivalent (by looping through the numbers 0-9 and checking if the 4 bits are equal to the 4 bits Player A has entered) and then performing the same operations as before.

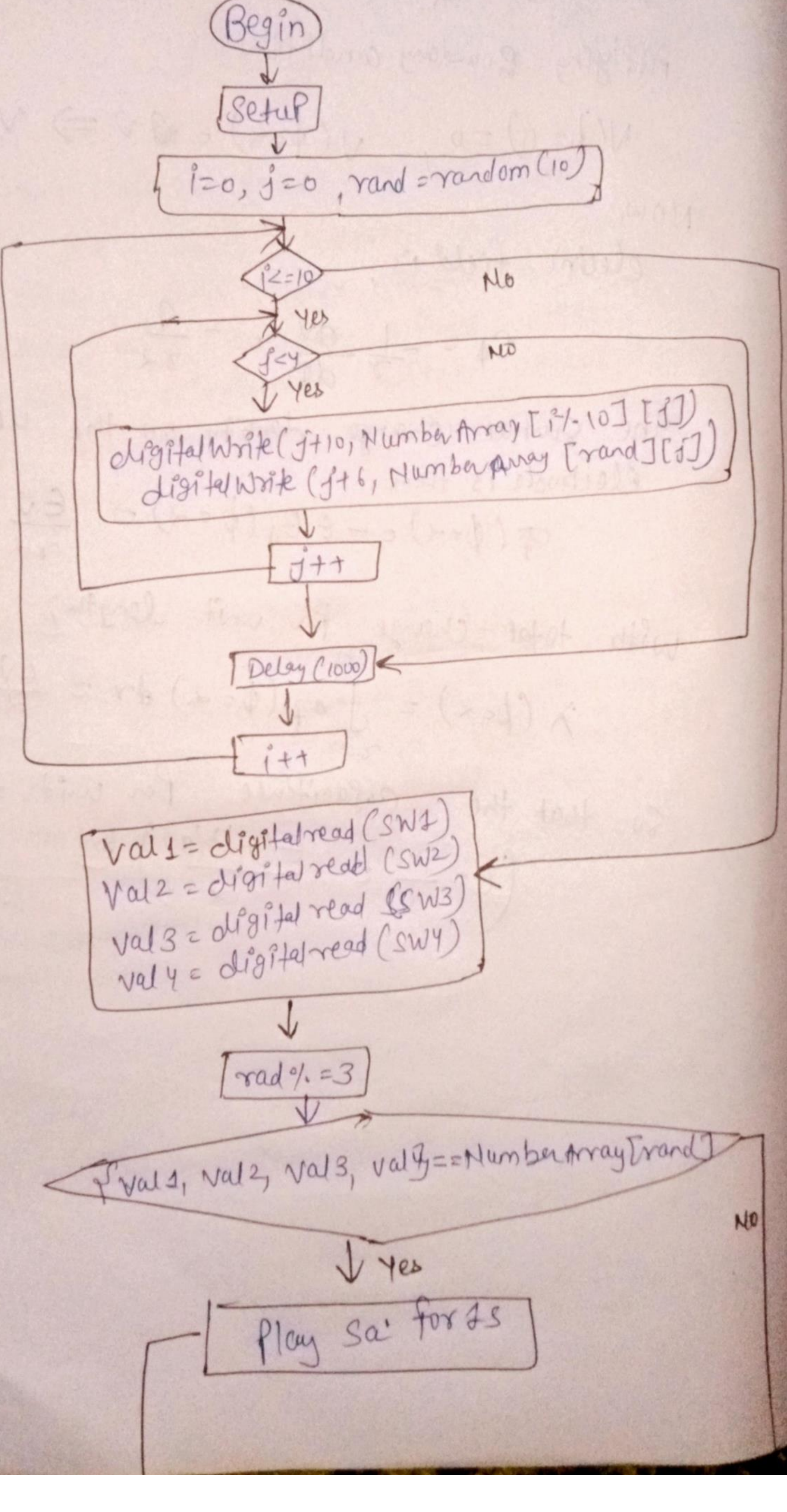
Module – 2b(Design a Game in TinkerCad Using the Circuit Elements of Module - 2a)

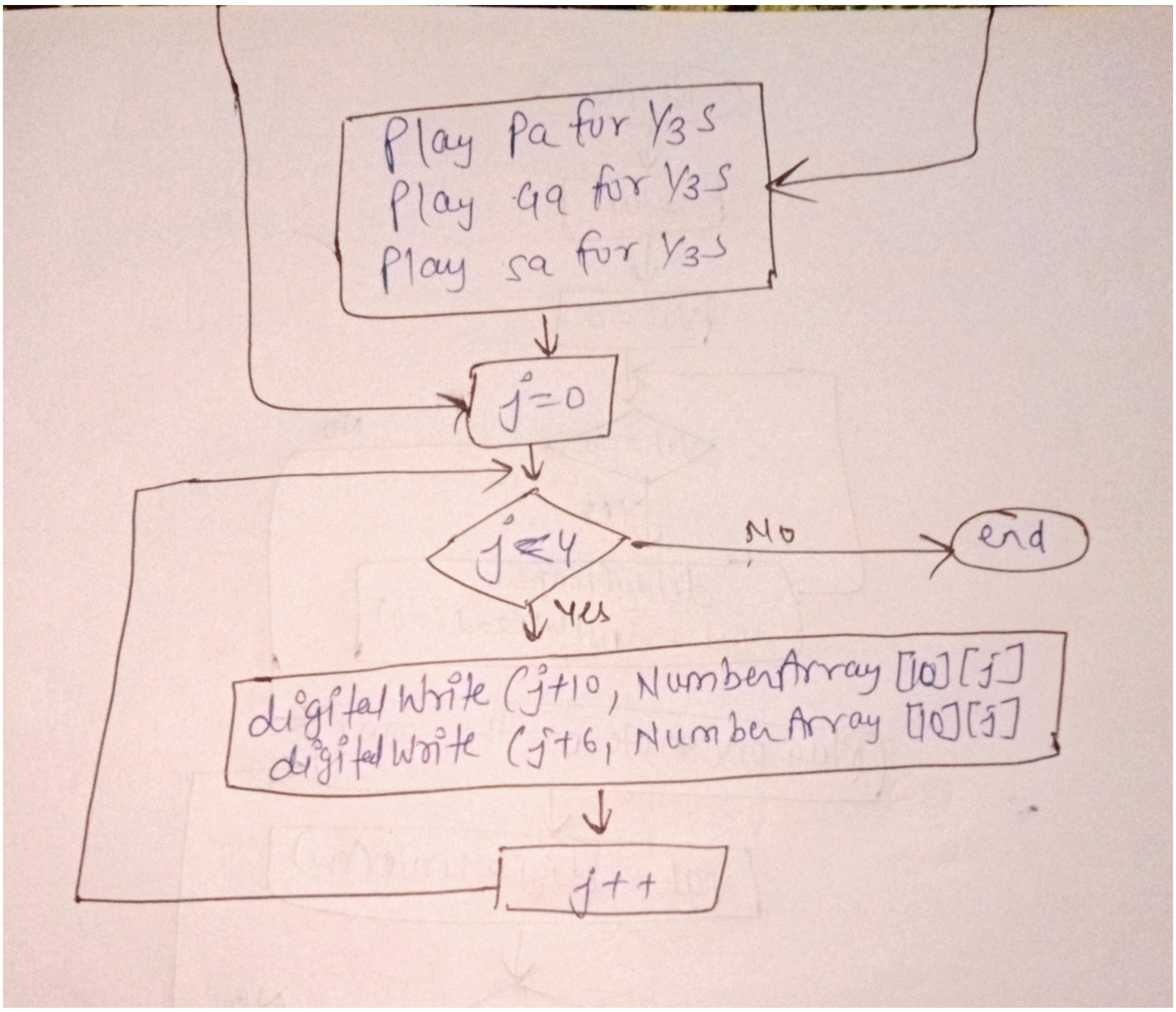
For this module, let us make a Collatz Game. This game is based on the famous unproven conjecture in Mathematics, the Collatz conjecture. The conjecture is concerned with sequences and is defined as follows: “start with any positive integer n. Then each term is obtained from the previous term as follows: if the previous term is even, the next term is one half of the previous term. If the previous term is odd, the next term is 3 times the previous term plus 1. The conjecture is that no matter what value of n, the sequence will always reach 1.”

Hence, starting with each positive number, we get a sequence of numbers leading to 1. In the Collatz Game the Player A gives the starting number. Player B then has to enter the number of steps required to go to 1, starting from the number given by Player A. There is a computer generated sequence of these number of steps required to reach 1 from positive numbers available at https://oeis.org/A006577. For the numbers 1-9, this sequence is 0, 1, 7, 2, 5, 8, 16, 3, 19. We should note that for an input of 0, there are no number of steps possible to reach 1. Hence, the answer in that case is said to be -1. Hence, in that case the player should enter the 2’s complement of -1. Since the maximum number in this sequence upto 9 is 19, we require 5 switches for player B to be able to enter the correct answer. Hence, we replace the 4-switch DIP in Module - 02.A by a 6-switch DIP. We take the inputs of these switches into pins 1-5. Hence, we define the input pins 5, 4, 3, 2 and 1 of the DIP switch as SW5, SW4, SW3, SW2 and SW1 respectively, using #define statements. The piezo-buzzer output pin is associated with pin 0 of the Arduino. In void setup(){...} , we have all the same statements as in Module - 02.A except for 1 extra statement for that 1 extra switch input coming from the DIP switch. The 2-D array int NumberArray[11][4] remains the same in this game as in Module - 02.A. In order to check whether Player B’s answer is correct or not, we define another 2-D array, int CollatzArray[10][5] containing the correct answers for each input. This array contains the binary equivalents of the number of steps required to reach 1 starting from the numbers 0-9.

Number	Pin 5	Pin 4	Pin 3	Pin 2	Pin 1
0	1	1	1	1	1
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	1	1	1
4	0	0	0	1	0
5	0	0	1	0	1
6	0	1	0	0	0
7	1	0	0	0	0
8	0	0	0	1	1
9	1	0	0	1	1

Flow chart


Begin
Setup
i=0, j=0 , rand = random (10)
i<=10
No
Yes
j<4
NO
Yes
digitalWrite (j+10, NumberArray [i%10] [j])
digitalWrite (j+6, NumberArray [rand] [j])
j++
Delay (1000)
i++
Val 1 = digitalread (SW1)
Val 2 = digitalread (SW2)
val 3 = digital read (SW3)
val 4 = digitalread (SW4)
rad % =3
{val 1, val 2, val 3, val 4} == NumberArray [rand]
NO
Yes
Play Sa' for 1s


Play Pa for 1/3 s
Play Ga for 1/3 s
Play sa for 1/3 s
j=0
j<4
No
end
Yes
digitalWrite (j+10, NumberArray [10][j]
digitalWrite (j+6, NumberArray [10][j]
j++

Arduino code

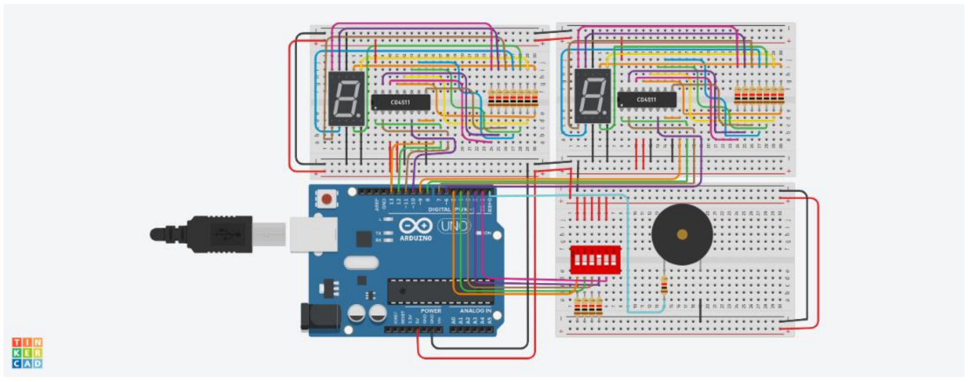
```
// C++ code

// define buzzer output as pin 0

#define BUZ 0

// define switch input as pins 1-5

#define SW5 1

#define SW4 2

#define SW3 3

#define SW2 4

#define SW1 5

// 2-D array representing 0 to 9 numbers

int NumberArray[11][4] = {{ 0, 0, 0, 0 },  // 0

                { 0, 0, 0, 1 },  // 1
```

```
                        { 0, 0, 1, 0 },  // 2
                        { 0, 0, 1, 1 },  // 3
                        { 0, 1, 0, 0 },  // 4
                        { 0, 1, 0, 1 },  // 5
                        { 0, 1, 1, 0 },  // 6
                        { 0, 1, 1, 1 },  // 7
                        { 1, 0, 0, 0 },  // 8
                        { 1, 0, 0, 1 },  // 9
                        { 1, 1, 1, 1 }}; // invalid input to stop displaying numbers on the 7-segment displays
// 2-D array representing correct answers
int CollatzArray[10][5] = {{ 1, 1, 1, 1, 1 },  // 0 : -1 = 31 (2's complement)
   { 0, 0, 0, 0, 0 },  // 1 : 0
   { 0, 0, 0, 0, 1 },  // 2 : 1
   { 0, 0, 1, 1, 1 },  // 3 : 7
   { 0, 0, 0, 1, 0 },  // 4 : 2
   { 0, 0, 1, 0, 1 },  // 5 : 5
   { 0, 1, 0, 0, 0 },  // 6 : 8
   { 1, 0, 0, 0, 0 },  // 7 : 16
   { 0, 0, 0, 1, 1 },  // 8 : 3
   { 1, 0, 0, 1, 1 }}; // 9 : 19
int val1, val2, val3, val4, val5; // variables to hold switch input values
void setup()
{
  // code for pins setup
  // random number display output pins initialization
  pinMode(6, OUTPUT); // making pin 6 as 7-segment display output
  pinMode(7, OUTPUT); // making pin 7 as 7-segment display output
  pinMode(8, OUTPUT); // making pin 8 as 7-segment display output
  pinMode(9, OUTPUT); // making pin 9 as 7-segment display output
  // timer count display output pins initialization
  pinMode(10, OUTPUT); // making pin 10 as 7-segment display output
  pinMode(11, OUTPUT); // making pin 11 as 7-segment display output
  pinMode(12, OUTPUT); // making pin 12 as 7-segment display output
  pinMode(13, OUTPUT); // making pin 13 as 7-segment display output
```

```
 // switch intput pins initialization
 pinMode(SW1, INPUT); // making pin 5 as first switch input
 pinMode(SW2, INPUT); // making pin 4 as second switch input
 pinMode(SW3, INPUT); // making pin 3 as third switch input
 pinMode(SW4, INPUT); // making pin 2 as fourth switch input
 pinMode(SW5, INPUT); // making pin 1 as fifth switch input
 pinMode(BUZ, OUTPUT); // making pin 6 as buzzer output
 randomSeed(analogRead(0)); // random seed
// code for the main game
 int i = 0, j = 0; // variables for looping over NumberArray
 int rand = random(10); // variable to store random number
 for (i = 0; i <= 10; i++) // loop for numbers
 {
  for (j = 0; j < 4; j++) // loop for bits of each number
  {
   digitalWrite(j + 10, NumberArray[i % 10][j]); // display count
   digitalWrite(j + 6, NumberArray[rand][j]); // display random number
  }
  delay(1000); // wait 1s before next assignment
 }
 val1 = digitalRead(SW1); // read first switch bit
 val2 = digitalRead(SW2); // read second switch bit
 val3 = digitalRead(SW3); // read third switch bit
 val4 = digitalRead(SW4); // read fourth switch bit
 val5 = digitalRead(SW5); // read fifth switch bit
// if input equals the CollatzArray value of the random number
 if ((val1 == CollatzArray[rand][0]) && (val2 == CollatzArray[rand][1]) && (val3 == CollatzArray[rand][2]) && (val4 ==
CollatzArray[rand][3]) && (val5 == CollatzArray[rand][4]))
 {
  tone(BUZ, 523.251, 1000); // play Sa'
  delay(1000); // wait for Sa' to have played
 }
 // if input does not equal the random number
 else
```

```
  {
    tone(BUZ, 391.995, 333); // play Pa
    delay(333); // wait for Pa to have played
    tone(BUZ, 329.628, 333); // play Ga
    delay(333); // wait for Ga to have played
    tone(BUZ, 261.626, 333); // play Sa
    delay(333); // wait for Sa to have played
  }
  // display nothing on both 7-segment displays
  for (j = 0; j < 4; j++) // loop for bits of each number
  {
    digitalWrite(j + 10, NumberArray[10][j]); // display nothing
    digitalWrite(j + 6, NumberArray[10][j]); // display nothing
  }
}
 void loop()
{
  // nothing here as we do not want to replay the game
}
```

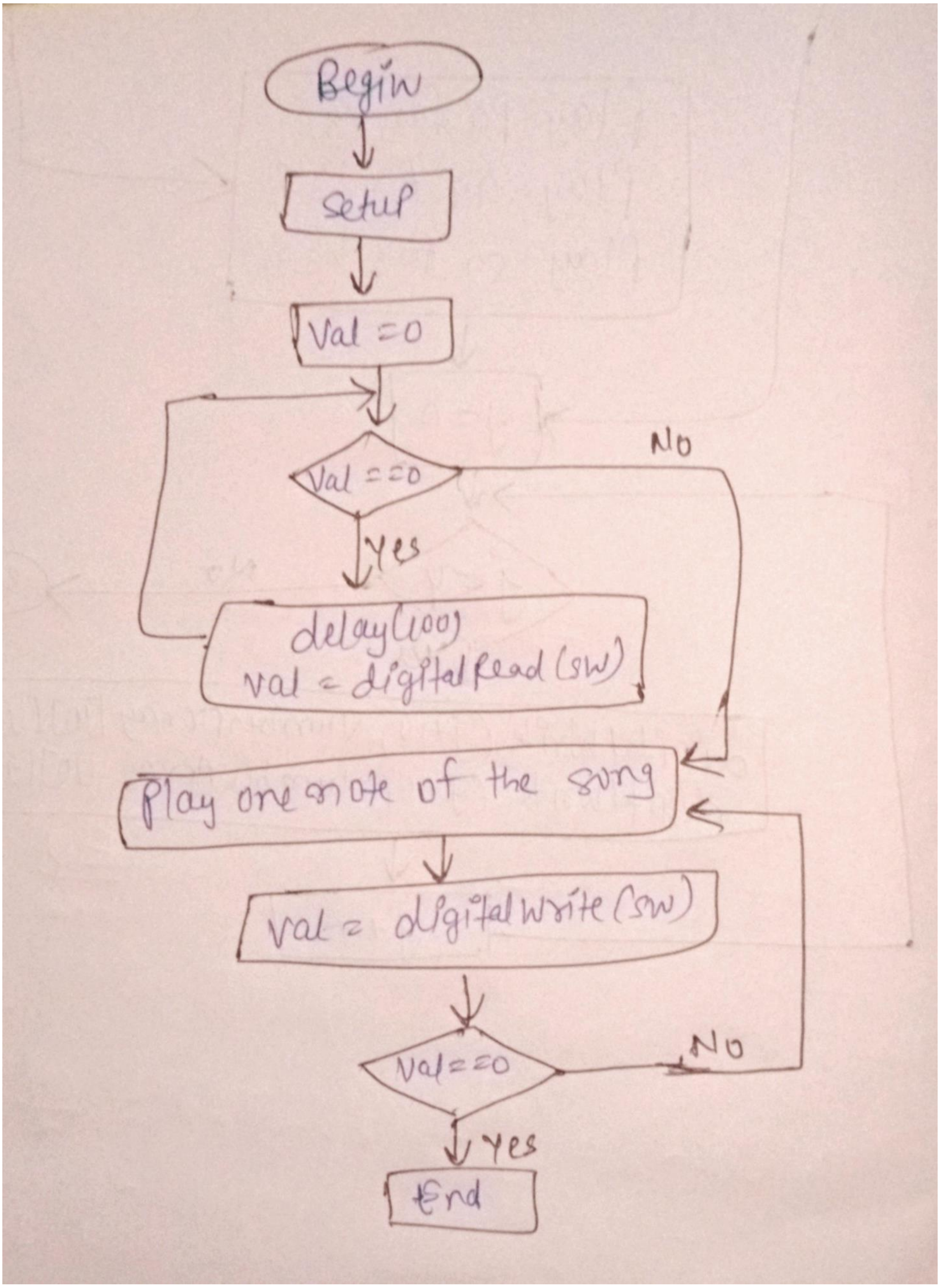
Circuit Diagram



Module – 2c(Use the Buzzer to Design an Automatic Music Player)

For this module, let us try to play the happy birthday tune using a buzzer.

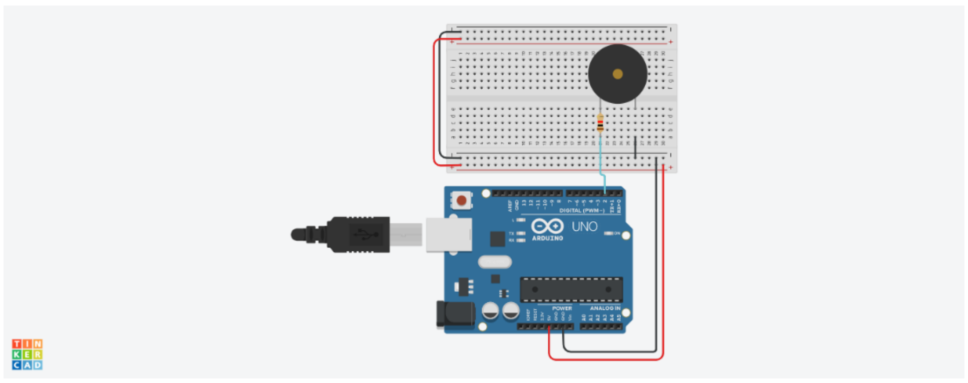
Flowchart



Arduino code

```
// C++ code
// define buzzer output as pin 2
#define BUZ 2
void setup()
{
  pinMode(BUZ, OUTPUT); // define BUZ pin as output
}
void loop()
{
  // happy birthday song
  // 1st stanza
  tone(BUZ, 130.813, 200); // play C note
  delay(300); // wait for one and a half duration
  tone(BUZ, 130.813, 100); // play C note
  delay(100); // wait for same duration
  tone(BUZ, 146.832, 400); // play D note
  delay(400); // wait for same duration
  tone(BUZ, 130.813, 400); // play C note
  delay(400); // wait for same duration
  tone(BUZ, 174.614, 400); // play F note
  delay(400); // wait for same duration
  tone(BUZ, 164.814, 400); // play E note
  delay(800); // wait for double the duration
  // 2nd stanza
  tone(BUZ, 130.813, 200); // play C note
  delay(300); // wait for one and a half duration
  tone(BUZ, 130.813, 100); // play C note
  delay(100); // wait for same duration
  tone(BUZ, 146.832, 400); // play D note
  delay(400); // wait for same duration
  tone(BUZ, 130.813, 400); // play C note
  delay(400); // wait for same duration
  tone(BUZ, 195.998, 400); // play G note
  delay(400); // wait for same duration
  tone(BUZ, 174.614, 400); // play F note
  delay(800); // wait for double the duration
  // 3rd stanza
  tone(BUZ, 130.813, 200); // play C note
  delay(300); // wait for one and a half duration
  tone(BUZ, 130.813, 100); // play C note
  delay(100); // wait for same duration
  tone(BUZ, 261.626, 400); // play C' note
  delay(400); // wait for same duration
  tone(BUZ, 220.000, 400); // play A note
  delay(400); // wait for same duration
  tone(BUZ, 174.614, 400); // play F note
  delay(400); // wait for same duration
  tone(BUZ, 164.814, 400); // play E note
  delay(400); // wait for same duration
  tone(BUZ, 146.832, 400); // play D note
  delay(400); // wait for same duration
  // 4th stanza
  tone(BUZ, 233.082, 200); // play Bb note
  delay(300); // wait for one and a half duration
  tone(BUZ, 233.082, 100); // play Bb note
  delay(100); // wait for same duration
  tone(BUZ, 220.000, 400); // play A note
  delay(400); // wait for same duration
  tone(BUZ, 174.614, 400); // play F note
  delay(400); // wait for same duration
  tone(BUZ, 195.998, 400); // play G note
  delay(400); // wait for same duration
  tone(BUZ, 174.614, 400); // play F note
  delay(800); // wait for double the duration
}
```

Circuit Diagram



References

[1] https://www.arduino.cc/reference/en/language/functions/ digital-io/pinmode/.

[2] https://store.arduino.cc/usa/arduino-uno-rev3.

[3] https://www.arduino.cc/reference/en.

[4] https://www.arduino.cc/reference/en/language/functions/ advanced-io/tone/.