IPiano - Electronic Piano With Arduino in Tinkercad (2 Octaves and Practice Mode)

by rheinnacyasimon

In this project, you will turn the Arduino into a Piano! This piano have 2 function, 2 octaves usual piano, and Practice mode, which user can practice hearing. This training are in the form of 3 type of question, Easy, Medium, and Hard. If the user presses one of these question buttons, the Piano will make sound of several notes, and the user must play the notes as they heard. If the user answers correctly, the green LED will light up. However, if the user answers incorrectly, the red LED will light up.

In this project, Arduino use push button as the notes input through analog pin A3-A5. For the speaker, we will use Piezo buzzer in Tinkercad.

Supplies:

Here's everything you will need :

1x Arduino Uno

28x pushbuttons

1x Resistor assortment - 3 x 470Ω,3 x 2kΩ, 3 x 200Ω, 3 x 400Ω, 3 x 800Ω, 3 x 1.6kΩ, 3 x 3.2kΩ, 3 x 6.4kΩ, 3 x 12.8kΩ

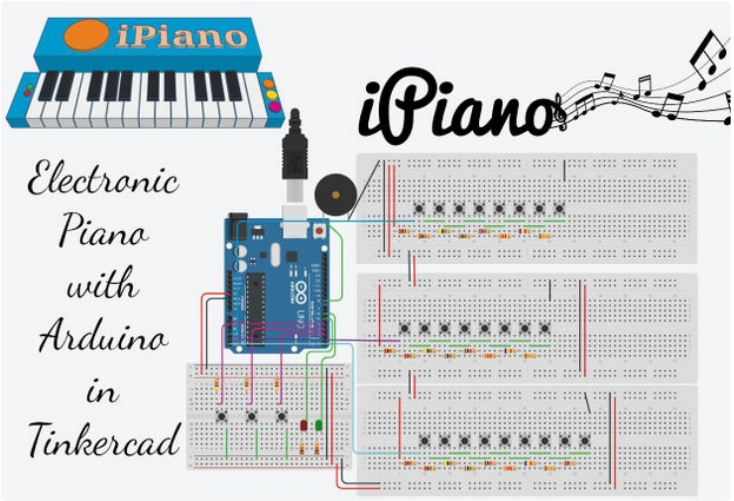
1x Piezo buzzer

1x Solderless breadboard

3x LED - red, green

In this project, you could build this simulated circuit along side the sample. If you want to build the physical circuit, gather up Arduino Uno board, USB cable, solderless breadboard, three LEDs, alike resistors, jumper, and piezo buzzer.

You can try to make your own iPiano using Tinkercad Circuits. You could also simulated this project from Tinkercad (free login required). Explore the sample circuit and build your own next to it! Tinkercad Circuits is a free browser program that lets you build and simulate circuits, build your own 3D design, and use Codeblocks.



Step 1: Build the Piano Pushbutton Circuit

First, we will build the circuit below in Tinkercad and make sure everything stick in its place. We will make 3 breadboard of this circuit that have exactly the same components to create the total of 25 notes or 2 octaves for the piano. We decided to split the piano into 3 breadboard to make the analog input value more wider.

You can look this example of our circuit.

https://www.tinkercad.com/embed/991E5SX6f5J?editbtn=1

Now, take the components needed to the workspace. We have Arduino Uno, Piezo Buzzer, 3 breadboards, some pushbutton, and some resistors. Put the Arduino Uno wherever you want. Now connect a wire from the 5V pin to the + in breadboard. Connect also a wire from the GND pin Arduino to - in breadboard. These two bring voltage to the breadboard. Then, connect the + and - on the side of breadboard to the other side of the breadboard.

After connect all the voltage to the three breadboard, we will put all the push button into the three breadboard. There are 3 breadboard, so we will divide it into 8 on the first and second breadboard, and 9 on the third breadboard.

Then, we will connect the pushbuttons and resistors. We will connect the right side of the first resistor with the left side of pushbutton, and connect it to the left side of second resistor. Connect again the right side of the second resistor with the left side of the resistor to the third resistor. We will do it until the last resistor on each breadboard.

If you can see the pattern, we put the first resistor 2000Ω to connect from Arduino to the pushbutton, then the second resistor that connect first pushbutton to other push button is from 200Ω, and go up 2 times from it. So we got 200, 400, 800, 1600, 3200, 6400, and finally 12800. The purpose of the value of the resistor made in pattern is to make the value in analog inputs will go up constantly. We will see the analog value pattern in Step 3. You can see the table in the image above.

$$Resistor\ Value = \frac{\sum_{n=0}^{n=n-1} Resistor \times Analog\ Value}{1023 - Analog\ Value}$$

Analog Value	Resistor Value	$\sum_{n=0}^{i=n-1} Resistor$	Lower Limit	Upper Limit
0	2000 (initiation)	0	0	50
93	200	2000	51	150
236	400	2200	151	300
421	800	2600	301	450
614	1600	3400	451	650
773	3200	5000	651	800
883	6400	8200	801	900
948	12800	14600	901	960
984	25600	27400	961	1000

Step 2: Add Test Button

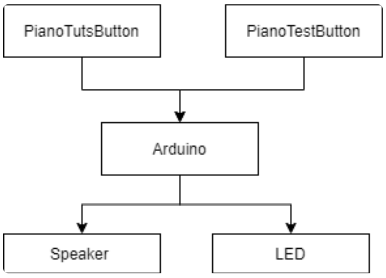
In the picture above, we can see the block diagram of our complete circuit. So, after making the piano button, we will make the test button and put it together with the piano button.

https://www.tinkercad.com/embed/jou29ljwN5n?editbtn=1

resistors for the pushbuttons, two LED (red and green), and two resistors for the LED. You can connect it like the simulation in Tinkercad above.

The digital pin 2, 3, and 4 will connect to the left side of the pushbutton, also resistors that connect from the + of the breadboard to the pushbuttons. The right side of the pushbutton will go down to negative side of breadboard.

For the LED, also we will connect it to the digital pin 5 and 6. The other side of LED will connect to resistor and go down to negative(-) of the breadboard. Don't forget



We already made the piano button. So, ignore the three breadboard with the 25 pushbuttons. Now, add a small breadboard for 3 type of practice mode, Easy, Medium, and Hard. We will need three pushbuttons, three

to connect the 5V to the positive(+) and GND to the negative(-), and also connect the + and - both side of the breadboard.

So, after making this test button, we will have a complete circuit!

https://www.tinkercad.com/embed/dMnxJknSiYW?editbtn=1

Step 3: Arduino Code for Playing Melody

For the third step, we will make the Arduino code for the melody of the piano. The aim is when we press the pushbutton, we can hear the exact same sound like real piano. In this piano, we will make the 2 octaves piano from C4 to C6.

Firstly, we will have to make an array for the frequencies from C4 to C6. Based on the trial and error to the real piano, we have conclude the frequencies in the picture above.

And now, for the values in the push button, we will make a variable called sensorValue. There are three breadboard so, we made 1 for each of them. sensorValue1 will read the analog value from analog pin A3, sensorValue2 will read the analog value from analog pin A4, and sensorValue3 will read the analog value from analog pin A5. After getting all the sensorValue by testing (based on the value of the resistor), we can get lowerLimit and upperLimit (on the picture above) for the condition used later.

Sensorvalue1 and sensorvalue2 only have 8 pushbuttons, but sensorvalue3 have 9 pushbuttons. So, sensorvalue1 and sensorvalue2 will use the table above until the value of 948, while sensorvalue3 use the table until 984.

After getting all required value, we will start coding. First things first, we will define all the value and pin we're using.

```
#define analogInPin1 A3
#define analogInPin2 A4
#define analogInPin3 A5
#define speaker 8
```

```
int sensorValue1 = 0;
int sensorValue2 = 0;
int sensorValue3 = 0;
```

After declaring variables, we also need to make an array of the tables above, like frequency, lowerLimit, and upperLimit.

```
int frequency[] = {262, 278, 294, 312, 330,
                   349, 371, 392, 416, 440,
                   467, 493, 523, 554, 587,
                   622, 660, 698, 739, 783,
                   830, 880, 932, 987, 1046};
```

```
int upperLimit[] = {50, 150, 300, 450, 650,
                    800, 900, 960, 1000};
```

```
int lowerLimit[] = {0, 51, 151, 301, 451,
                    651, 801, 901, 961};
```

Now, we will go on to the setup function. In this function we will set the output to speaker. You can print the sensor value also by putting the Serial.begin syntax on the setup. But the serial monitor print is optional.

```
void setup() {
  pinMode (speaker, OUTPUT);
  Serial.begin(9600);
}
```

And now, we will go to the piano melody function. We called it readMelody. Firstly, we will set to read the sensorValue1 in analogInPin1. Then, there will be a loop for, with i variables. We set the parameter of i start from 0, smaller than 8 (as much as the pushbutton). In this for loop, there will be a condition. If the value of sensorvalue1 in the range of lower limit and upper limit, then the piano will play the tone in the speaker. You can see the code below.

```
void readMelody() {
sensorValue1 = analogRead(analogInPin1);
  for(int i = 0; i<8; i++)
  {
    if (sensorValue1 < upperLimit[i]  && sensorValue1 >= lowerLimit[i]){
      tone(speaker,frequency[i],60);
    }
  }
}
```

Is it done yet? A little bit touch up and we're finished!

You can see that the sensorValue1 only read analogInPin1 which is the first breadboard. But, we have 3 breadboard, so we will copy this code in the readMelody function, and have almost the same code. What makes it different is just the read of analogInPin2 or analogInPin3, the condition of the sensorValue2 or sensorValue3, and also the parameter of i in the third breadboard - third breadboard have 9 pushbuttons).

So, here is the final code of function readMelody.

```
void readMelody() {
sensorValue1 = analogRead(analogInPin1);
  for(int i = 0; i<8; i++)
  {
    if (sensorValue1 < upperLimit[i]  && sensorValue1 >= lowerLimit[i]){
      tone(speaker,frequency[i],60);
    }
  }
  sensorValue2 = analogRead(analogInPin2);
  for(int i = 0; i<8; i++)
  {
    if (sensorValue2 < upperLimit[i]  && sensorValue2 >= lowerLimit[i]){
      tone(speaker,frequency[i+8],60);
    }
  }
  sensorValue3 = analogRead(analogInPin3);
  for(int i = 0; i<9; i++)
  {
    if (sensorValue3 < upperLimit[i]  && sensorValue3 >= lowerLimit[i]){
      tone(speaker,frequency[i+16],60);
    }
  }
}
```

Wait, wait.. why the tone syntax have this weird i+number thing? If you notice that, you have a critical thinking! So, basically, we will play the tone by calling the frequency array. For example, if we pressed the first button on the first breadboard, we will heard the C4 (frequency[0] = 262). Imagine the number 0 is replaced by i. If we pressed the first pushbutton in the second breadboard, the i variable will start looping in the for syntax from 0 again. But, it will play the note G#4. So, we can handle that by giving extra code in frequency[i+ number of push button earlier] in the second and third for loop.

After the void setup, we will go on to void loop. In this function we will print the sensorvalues, and also we will call the readMelody function.

```
void loop() {
    readMelody();

  Serial.print("Value1 = ");
  Serial.println(sensorValue1);
  Serial.print("Value2 = ");
  Serial.println(sensorValue2);
  Serial.print("Value3 = ");
  Serial.println(sensorValue3);
}
```

Yeay!! We're finished making the melody part of the Piano

https://www.tinkercad.com/embed/0UZTHJUvunR?editbtn=1

Notes	Frequency		
C4	261	C5	523
C#4	278	C#5	554
D4	294	D5	587
D#4	312	D#5	622
E4	330	E5	660
F4	349	F5	698
F#4	371	F#5	739
G4	392	G5	783
G#4	416	G#5	830
A4	440	A5	880
A#4	467	A#5	932
B4	493	B5	987
		C6	1046

Value	Lower Limit	Upper Limit
0	0	50
93	51	150
236	151	300
421	301	450
614	451	650
773	651	800
883	801	900
948	901	960
984	961	1000

Step 4: Arduino Code for Generating Question

In this step we're going to explain how to generate the question, the moment the user presses the test button. There are 3 types of test button which are easy, medium, hard. First, we are going to discuss about the easy mode.

```
void easyquestion(){
  for(int j =0;j<4;j++){
   digitalWrite(speaker,HIGH);
   int randomfrequency = frequency[random(7)];
   easyques[j]=randomfrequency;
   tone(speaker,randomfrequency,300);
   digitalWrite(speaker,LOW);
   delay(500);
   Serial.println(easyques[j]);
  }
 }
```

We are using an array named easyques with 'j' number of elements which is 4, based on the for function. Then a random frequency is saved by the array. The maximum random value that system are taking from the frequency arrays are 8 (0-7) which are the first 8 notes of the piano(which is the first breadboard). After that, using a function called tone, the note saved in the array is played for the user to hear the notes.

We did the same thing with the other 2 levels which are medium and hard. But the difference of every level is the maximum number of possibilities the notes chosen. Here is the medium question code:

```
void medquestion(){
  for(int j =0;j<4;j++){
     digitalWrite(speaker,HIGH);
     int randomfrequency = frequency[random(15)];
     medques[j]=randomfrequency;
     tone(speaker,randomfrequency,300);
     digitalWrite(speaker,LOW);
     delay(500);
     Serial.println(medques[j]);
   }
}
```

Pay attention at the fourth line of the code, the number randomized is not 7 but its 15. Therefore it has more possibilities than easy questions. And there are 4 numbers of question notes because at the second line of the coding we made a loop that repeat four times. Then similar with the easy question, I use the tone function to tell the piezo to sound the frequency randomized.

Next, we are going to explain the difference in hard questions. Here is the code:

```
void hardquestion(){
   for(int j =0;j<4;j++){
     digitalWrite(speaker,HIGH);
     int randomfrequency = frequency[random(24)];
     hardques[j]=randomfrequency;
     tone(speaker,randomfrequency,300);
     digitalWrite(speaker,LOW);
     delay(500);
     Serial.println(hardques[j]);
  }
}
```

Notice in the fourth line the number of random is 24 it means that there are more possibilities a note can be picked and saved in the array. That is the only difference in the code between easy, medium and hard. And based on the for function used we repeat the function four times and that's why we will have 4 questions too for the hard mode.

Step 5: Arduino Code for Checking Answer

You can see in the picture above is our flowchart of the iPiano. So now, in this step we will explain how to check the user's answers. Basically, we are just comparing the elements of the question array with the answer array. Here is the code for easy check:

```
void easyquestioncheck(){
  for(k =0;k<4;k++){
    sensorValue1=1023;
     while (sensorValue1==1023){
     sensorValue1 = analogRead(analogInPin1);
```

We are going to use a for function to input the answer 4 times, and the while function is used to hold the system until the user input an answer. Because the sensorValue1 always has a value of 1023 if there are no inputs, therefore the system won't get out from the function until the user input a value in the first 8 notes of the piano.

```
    for(i = 0; i<8; i++)
    {
      if (sensorValue1 < upperLimit[i]  && sensorValue1 >= lowerLimit[i]){
        tone(speaker,frequency[i],60);
        easyanswer[k]=frequency[i];
      }
    }
  }// brackets for the while function
```

The sensor value received from the previous function is run in this function to produce the sound in piezo and the frequency produced in piezo is saved inside the array of easyanswer. So we don't save the sensorValue1 but we saved the frequency produced by inputting the value of sensorValue1 to the code. So in the explanation of the next code basically we are just checking the frequency value, do they match each other?

```
if(easyanswer[k] == easyques[k]){
    digitalWrite(hijau,HIGH);
    delay(200);
    digitalWrite(hijau,LOW);
}
```

Then we use an if function to compare the value of elements inside easyanswer and easyques array, the array is compared one by one according to the variable k which the variable used in the previous for function. If the value of element of easyanswer is the same with the value of easyques then, LED green will turn on for 200ms and then it will turn off.

```
if(easyanswer[k] != easyques[k]){
    digitalWrite(merah,HIGH);
    delay(200);
    digitalWrite(merah,LOW);
    }
  }  // close bracket for the first for loop
} // close the bracket for the void easyquesioncheck
```

This If function is the condition when the user input the wrong answer because in this if function the system compare the element of easyanswer and the element of easyques, if they are not the same then LED red will turn on and turn off in 200ms.

Here is the complete code of easy question check function :

```
void easyquestioncheck(){
 for(k =0;k<4;k++){
  sensorValue1=1023;
    while (sensorValue1==1023){
    sensorValue1 = analogRead(analogInPin1);
     for(i = 0; i<8; i++)
     {
       if (sensorValue1 < upperLimit[i]  && sensorValue1 >= lowerLimit[i]){
         tone(speaker,frequency[i],60);
         easyanswer[k]=frequency[i];
       }
     }
    }
  if(easyanswer[k] == easyques[k]){
  digitalWrite(hijau,HIGH);
  delay(200);
  digitalWrite(hijau,LOW);
  }
  if(easyanswer[k] != easyques[k]){
  digitalWrite(merah,HIGH);
  delay(200);
  digitalWrite(merah,LOW);
  }
 }
}
```

Next is the code to check medium question, the difference between easy and medium are the number of sensor values in easy is only sensorValue1 while in medium there are 2 sensor values which are sensorValue1 and sensorValue2, therefore it's going to be a little tricky here. Why did medium question has 2 sensor values this is caused by the number of possibilities of the random notes produced. In easy question there are only 8 possibilities while in medium question there are 16 possibilities. Here is the code:

```
void medquestioncheck(){
 for(k =0;k<4;k++){
  sensorValue1=1023;
  sensorValue2=1023;
    while (sensorValue1==1023&&sensorValue2==1023){
```

In this function we have 2 sensor value and we use while function to wait for the user to give input to the piano, either one of the sensor values changed will break the loop.

```
     sensorValue1 = analogRead(analogInPin1);
     for(i = 0; i<8; i++)
     {
       if (sensorValue1 < upperLimit[i]  && sensorValue1 >= lowerLimit[i]){
         tone(speaker,frequency[i],60);
         medanswer1[k]=frequency[i];
       }
```

After the program continued from the while loop is saved, the value of the sensorValue1 in this function inside an array named medanswer1.

```
      sensorValue2 = analogRead(analogInPin2);
      for(int i = 0; i<8; i++)
      {
        if (sensorValue2 < upperLimit[i]  && sensorValue2 >= lowerLimit[i]){
          tone(speaker,frequency[i+8],60);
          medanswer2[k]=frequency[i+8];
        }
      }
    }
  }
```

Since in medium mode we need 2 sensor values, therefore we make another function to save the sensorValue2 inside an

array named medanswer2.

```
if(medanswer1[k] == medques[k]|| medanswer2[k] == medques[k]){
   digitalWrite(hijau,HIGH);
   delay(200);
   digitalWrite(hijau,LOW);
}
```

Using the condition above which is if either one of the array has the same value with the array that contain the question, it means the user answer the question correctly, and the piano will turn on the green LED for 200ms and turn it back off.

```
if(medanswer1[k] != medques[k]&& medanswer2[k] != medques[k]){
   digitalWrite(merah,HIGH);
   delay(200);
   digitalWrite(merah,LOW);
   }
  }
}
```

While if both answer didn't have the same value with the question, the piano will turn on the red LED and turn it off after 200ms.

Here is the complete code of the medium question check :

```
void medquestioncheck(){
  for(k =0;k<4;k++){
    sensorValue1=1023;
    sensorValue2=1023;
      while (sensorValue1==1023&&sensorValue2==1023){
        sensorValue1 = analogRead(analogInPin1);
        for(i = 0; i<8; i++)
        {
          if (sensorValue1 < upperLimit[i]  && sensorValue1 >= lowerLimit[i]){
            tone(speaker,frequency[i],60);
            medanswer1[k]=frequency[i];
          }
          sensorValue2 = analogRead(analogInPin2);
          for(int i = 0; i<8; i++)
          {
            if (sensorValue2 < upperLimit[i]  && sensorValue2 >= lowerLimit[i]){
              tone(speaker,frequency[i+8],60);
              medanswer2[k]=frequency[i+8];
            }
          }
        }
      }
    if(medanswer1[k] == medques[k]|| medanswer2[k] == medques[k]){
      digitalWrite(hijau,HIGH);
      delay(200);
      digitalWrite(hijau,LOW);
    }
    if(medanswer1[k] != medques[k]&& medanswer2[k] != medques[k]){
      digitalWrite(merah,HIGH);
      delay(200);
      digitalWrite(merah,LOW);
    }
  }
}
```

And our last step is to make the checking algorithm of the hard question, this time its more complicated since it has 3 sensor values. Here is the code:

```
void hardquestioncheck(){
 for(k =0;k<4;k++){
   sensorValue1=1023;
   sensorValue2=1023;
   sensorValue3=1023;
    while (sensorValue1==1023&&sensorValue2==1023&&sensorValue3==1023){
```

As we have explained in the medium question why we use 2 sensor values, it is the same in the hard mode because the note randomized in this mode is not 16 anymore but it is 25 notes. And we used 3 analog pin that's why we need 3 sensor values.

```
sensorValue1 = analogRead(analogInPin1);
for(i = 0; i<8; i++){
   if (sensorValue1 < upperLimit[i]  && sensorValue1 >= lowerLimit[i]){
     tone(speaker,frequency[i],60);
     hardanswer1[k]=frequency[i];
   }
```

The code above is used to save the frequency produced by sensorvalue1 in hardanswer1 and play the frequency at the piezo using tone function.

```
sensorValue2 = analogRead(analogInPin2);
 for(int i = 0; i<8; i++)
 {
   if (sensorValue2 < upperLimit[i]  && sensorValue2 >= lowerLimit[i]){
     tone(speaker,frequency[i+8],60);
     hardanswer2[k]=frequency[i+8];
   }
 }
```

While this code is used to save the frequency produced by sensorvalue2 inside a array named hardanswer2.

```
sensorValue3 = analogRead(analogInPin3);
 for(int i = 0; i<9; i++)
 {
   if (sensorValue3 < upperLimit[i]  && sensorValue3 >= lowerLimit[i]){
     tone(speaker,frequency[i+16],60);
     hardanswer3[k]=frequency[i+16];
   }
 }
 }
}
```

Then the third frequency produced by sensorValue 3 is saved in an arrays named hardanswer3.

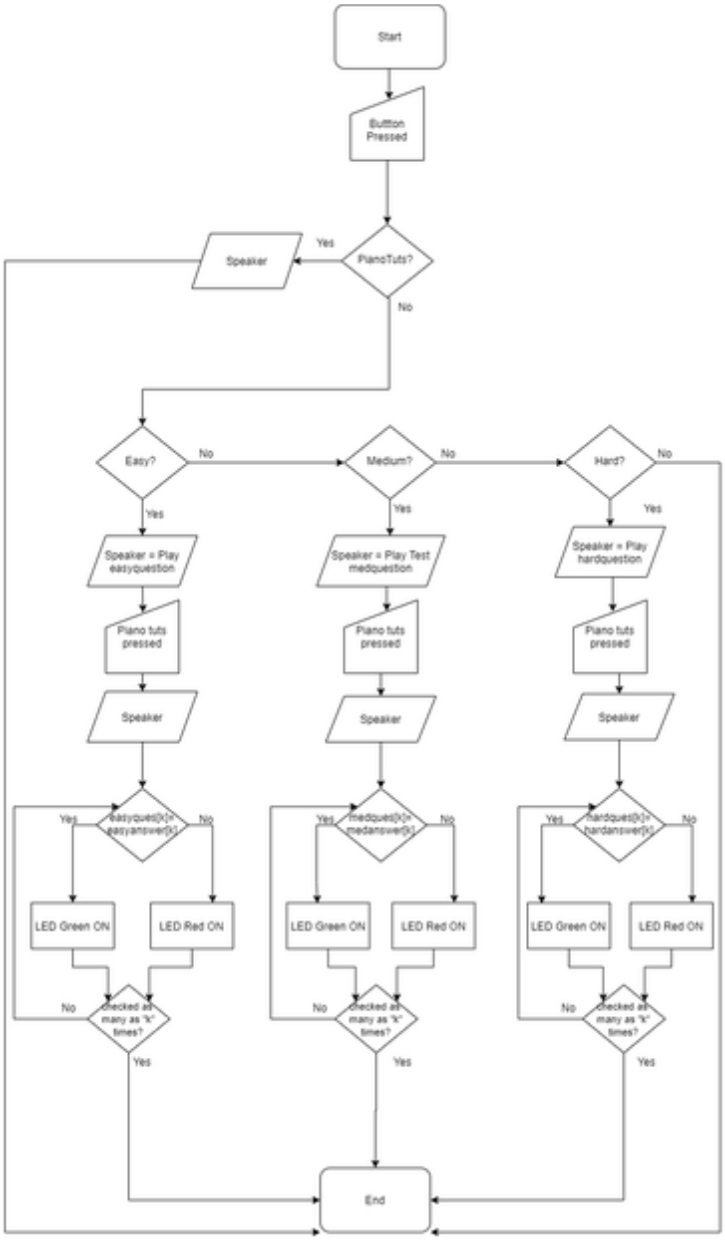
```
if(hardanswer1[k] == hardques[k]|| hardanswer2[k] == hardques[k]|| hardanswer3[k] == hardques[k]){
   digitalWrite(hijau,HIGH);
   delay(200);
   digitalWrite(hijau,LOW);
   }
```

After we save all the frequency inputted by the user using sensorValues we compare the answer with the question, if 1 of the answer match the question than the user guess the answer correctly and green LED will turn on for 200ms and turn off.

```
if(hardanswer1[k] != hardques[k]&& hardanswer2[k] != hardques[k]&& hardanswer3[k] != hardques[k]){
   digitalWrite(merah,HIGH);
   delay(200);
   digitalWrite(merah,LOW);
   }
  }
}
```

But if none of the answer has the same value with the question it means that the red LED will turn on and off after 200ms. Here is the complete code of the hard question check :

```
void hardquestioncheck(){
  for(k =0;k<4;k++){
   sensorValue1=1023;
   sensorValue2=1023;
   sensorValue3=1023;
     while (sensorValue1==1023&&sensorValue2==1023&&sensorValue3==1023){
       sensorValue1 = analogRead(analogInPin1);
       for(i = 0; i<8; i++)
       {
         if (sensorValue1 < upperLimit[i]  && sensorValue1 >= lowerLimit[i]){
           tone(speaker,frequency[i],60);
           hardanswer1[k]=frequency[i];
         }
         sensorValue2 = analogRead(analogInPin2);
         for(int i = 0; i<8; i++)
         {
           if (sensorValue2 < upperLimit[i]  && sensorValue2 >= lowerLimit[i]){
             tone(speaker,frequency[i+8],60);
             hardanswer2[k]=frequency[i+8];
           }
         }
         sensorValue3 = analogRead(analogInPin3);
         for(int i = 0; i<9; i++)
         {
           if (sensorValue3 < upperLimit[i]  && sensorValue3 >= lowerLimit[i]){
             tone(speaker,frequency[i+16],60);
             hardanswer3[k]=frequency[i+16];
           }
         }
       }
     }
   if(hardanswer1[k] == hardques[k]|| hardanswer2[k] == hardques[k]|| hardanswer3[k] == hardques[k]){
     digitalWrite(hijau,HIGH);
     delay(200);
     digitalWrite(hijau,LOW);
   }
   if(hardanswer1[k] != hardques[k]&& hardanswer2[k] != hardques[k]&& hardanswer3[k] != hardques[k]){
     digitalWrite(merah,HIGH);
     delay(200);
     digitalWrite(merah,LOW);
   }
  }
}
```



Step 6: Finally...

So.... how it is?? Great! You have finished your own piano! You can see the simulation on the youtube video above!

Here is our example of the complete iPiano!

https://www.tinkercad.com/embed/cYZfmpm8NZF?editbtn=1

Is your piano doing great? Hope you guys can make your own and develop it better!

Conclusions :

1. iPiano can play the same frequency as the real piano, but we still have to fix the voice clarity, also this is only a simulation.

2. iPiano can perform practice mode as it planned.

3. iPiano can only give 3 question in each level, so it fits for the beginners.

Writer :

Arsenius Davin (Computer Engineering)

Rheinnacya Ivanne (Computer Engineering)

https://youtu.be/TXHXzEv8ysU