**Can you feel my heart beat?**

|  |
| --- |
| Team : AED ( 3 조 )  Name : Min-sang Kim 20170462  Jae-hwan Kim 20170476  Ye-ram Lee 20170521  Ji-eun Jeong 20170537  Hye-jeong Kim 20170482  Class : 공학설계입문  ( Tue 12:00 ~ 1:15, Thu 9:00 ~ 10:15 )  Instructor : Dr. Kwang-Bock You  Date : 2017. 12. 15. |

Content

1. Introduction of item
2. Hardware
3. Source Code

* Jarduino-BT-mini Setup source code with sketch
* Jarduino-BT-mini main source code with sketch
* Arduino UNO R3 source code with sketch
* Raspberry pi 3 source code with python

1. Expected effect

Figure content

1. Figure 1 : Materials of this project
2. Figure 2 : Immediate constituent
3. Figure 3 : How it Works
4. Figure 4 : circuit demo version
5. Figure 5 : Completed version
6. Figure 6 : Final drawing of the circuit
7. Figure 7 : circuit diagram
8. Figure 8 : App inventor

[[1]](#footnote-1)Rough description of our project

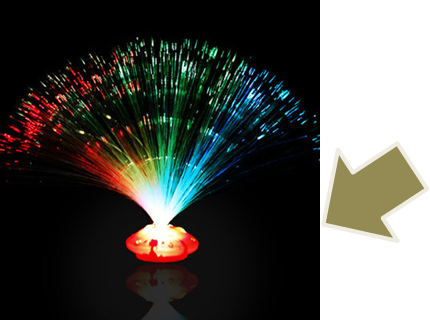
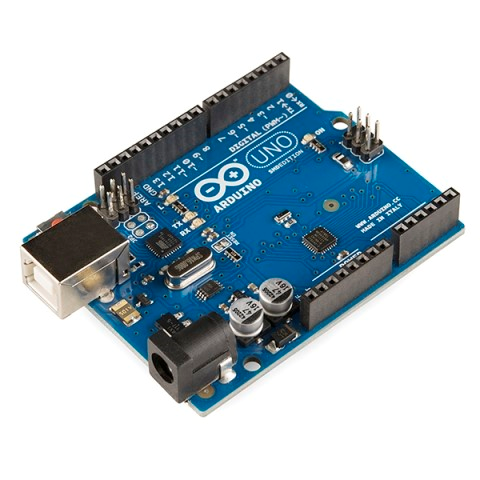
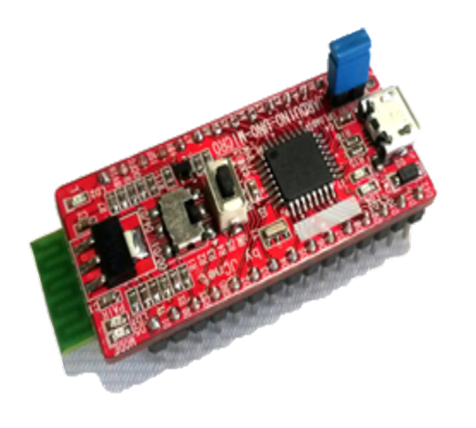
* There are many Mood Light which is only brightened, but our project <Can You Feel My Heartbeat?> is controlled by the heartbeat.
* We can listen to music which is similar to our BPM and suits our psychological state.

1. Differences from conventional product

* We integrated some functions of each product.
* Through BPM you measured, it plays music and differs the brightness and color of light. Also, frequency of light is similar to our BPM.
* Moreover, we used Raspberry Pi. It has infinite possibilities so that we can create more various functions even after this project.

1. What parts we used?

* We replaced the role of light material to Optical fiber. It has an advantage that there’s low energy loss if we use it, so our intent to use it was to display the heartbeat visually.
* We used Heart rate sensor to measure our BPM per minute.
* Basically, we also used Arduino. It receives the BPM and controls the LED light.
* Additionally, Raspberry Pi was used. It plays music of BPM similar to heart rate. Also, it has extensibility features so that we can create more.
* Jarduino uno BT mini was used to receive BPM and sending it to the Arduino.

<Figure 1> Materials of this project

[[2]](#footnote-2)

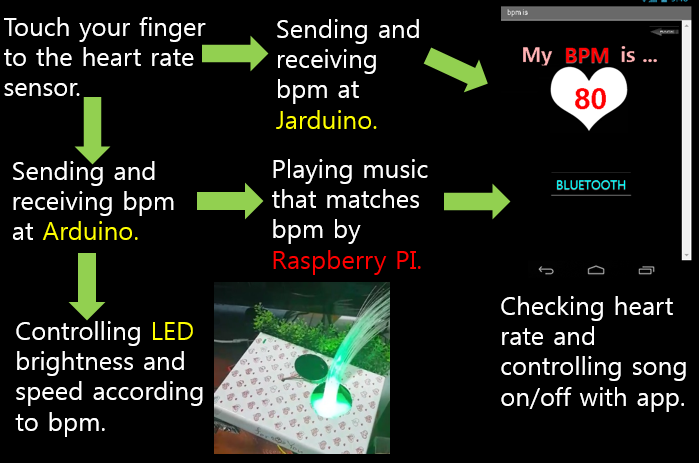
[[3]](#footnote-3)Here is a picture of the components we actually used.

전자기기이(가) 표시된 사진

높은 신뢰도로 생성된 설명 < Figure 2. Immediate constituent>

We connected the heart rate sensor to the Jarduino, connected the Jarduino and Arduino, and connected the Arduino and the Raspberry PI. The LEDs was connected to the Arduino, and the speaker was connected to the raspberry pI.

Accordingly, the work principle of our work is as follows.



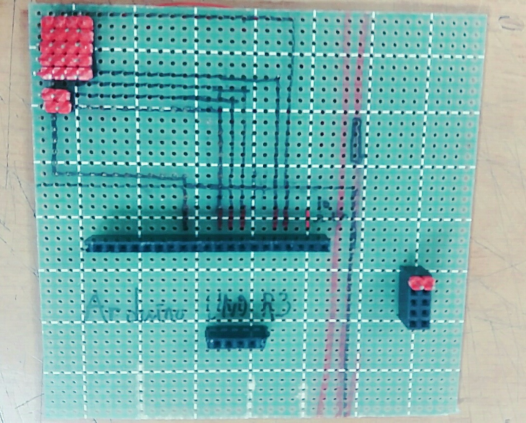
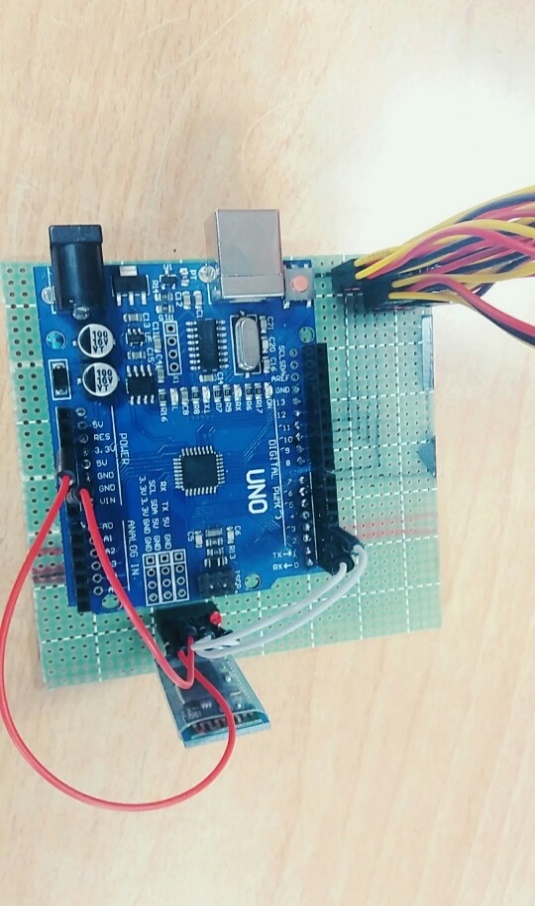
< Figure 3. How it Works>

First, you first measure your BPM by touching your finger to the heart rate sensor, and then transmit and receive it to the Jarduino and Arduino. Arduino controls the brightness and speed of the LEDs according to the BPM, and sends the BPM to Raspberry PI to play the music that matches the BPM. We have created an app that allows you to check your heart rate and control music on & off, which can be done if only Bluetooth is connected.

In addition to this project, we used Raspberry PI to make IOT technology an easy-to-use environment. We controlled the Linux environment from Raspberry PI to Python. We created a program that when it received the BPM value from Arduino, automatically plays music similar to the BPM value.

[[4]](#footnote-4)

[[5]](#footnote-5)

We separated four tri-color LEDs and five tri-color LEDs, and connected them in parallel. The four LEDs and five tri-color LEDs were controlled at different time intervals. We turned the simulations on a simulation site called Tinkercad and got the effect of controlling each LED with maximum brightness through the characteristics of the circuit.

< Figure 4. circuit demo version> < Figure 5. Completed version>

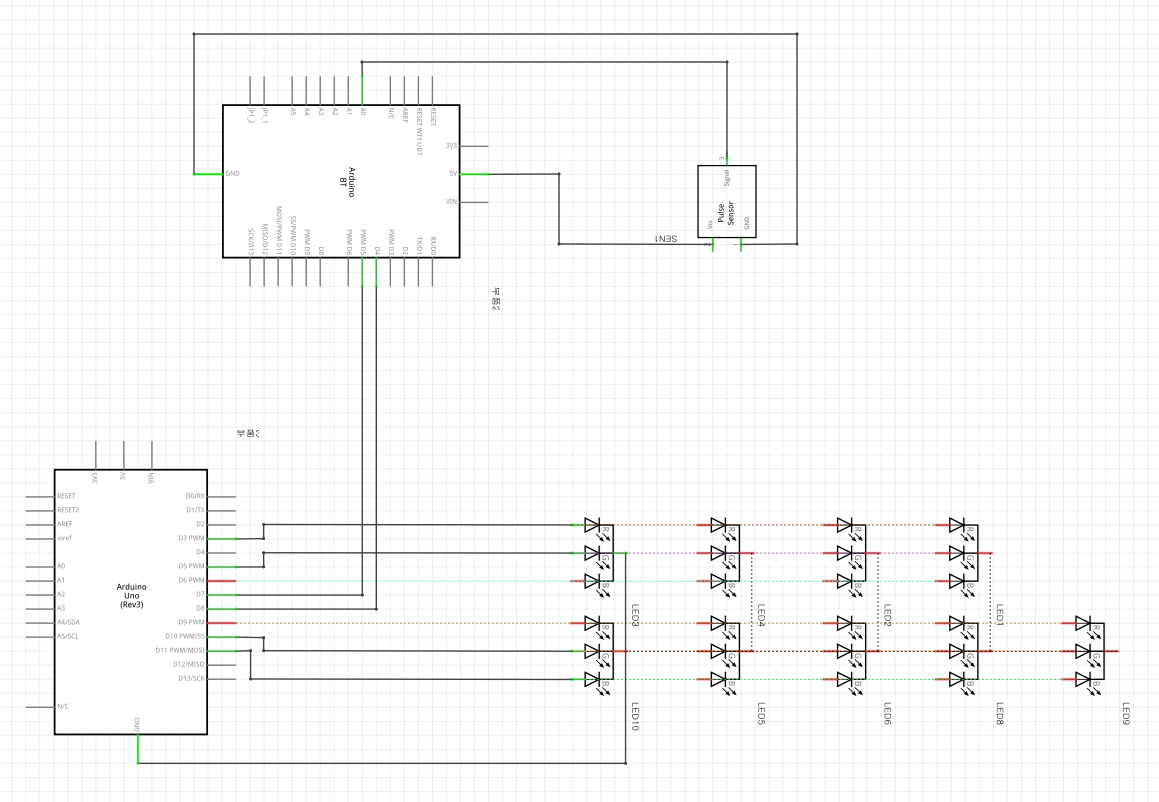
사람, 실내, 컴퓨터, 테이블이(가) 표시된 사진

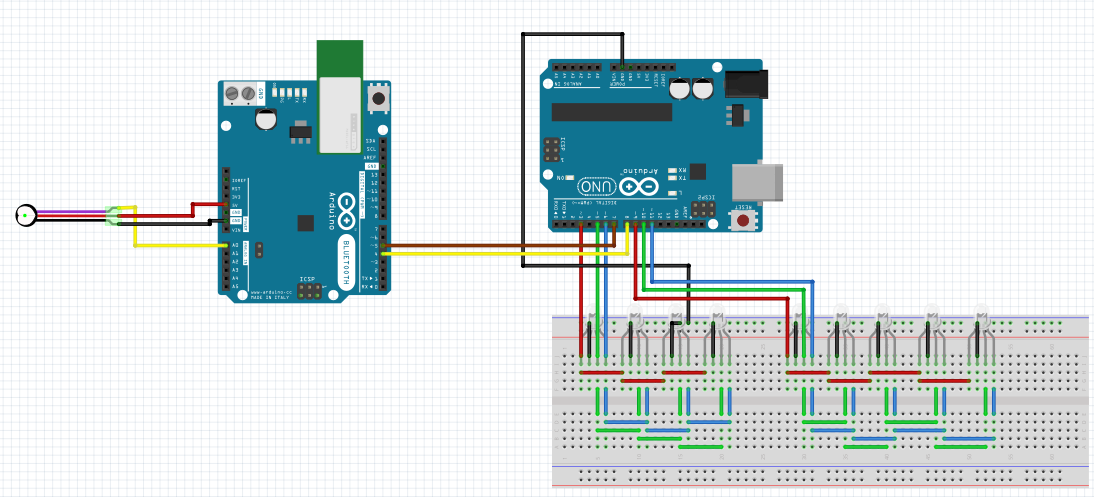
매우 높은 신뢰도로 생성된 설명

< Figure 6 - Final drawing of the circuit >

On the other hand, red pin is connected to 9, 11, Green pin is connected 5, 10, and Blue pin is connected to 6, 3 in Arduino. Unlike other circuit diagrams, the advantage of this circuit is that it convenient to connect the other cable at this bread board

The following figures are circuit diagrams.[[6]](#footnote-6)





< Figure 7 – circuit diagram >

[[7]](#footnote-7)

**Source code of Jarduino-BT-mini Bluetooth Setup**

|  |
| --- |
| 1. #include<SoftwareSerial.h> 2. SoftwareSerial BTSerial(4, 7); 3. void setup() { 4. Serial.begin(9600); 5. BTSerial.begin(38400); 6. } 7. void loop() { 8. if(Serial.available()) 9. BTSerial.write(Serial.read()); 11. if(BTSerial.available()) 12. Serial.write(BTSerial.read()); 13. } |

Line 7 : Set up Bluetooth baud speed to 38400 in Jarduino-BT-mini.

Line 11 : If some serial value exist, it will be true. If not, it will be false.

Line 12 : When it receive from serial monitor, it will send Bluetooth a value that read from serial monitor.

Line 14 : If some value exist in Bluetooth, it will be true. If not, it will be false.

Line 15 : When it receive from Bluetooth, it will send serial monitor a value that read from bluetooth.

**Jarduino-BT-mini Sketch Code**

[[8]](#footnote-8)

PulseSensorAmped\_Arduino\_1dot4.ino

|  |
| --- |
| 1. #include<SoftwareSerial.h> 2. SoftwareSerial BTSerial(4 ,7); 3. SoftwareSerial Bluetooth(0 ,1); 4. int pulsePin = 0; 5. int blinkPin = 13; 6. int fadePin = 5; 7. int fadeRate = 0; 8. volatile int BPM; 9. volatile int Signal; 10. volatile int IBI = 600; 11. volatile boolean Pulse = false; 12. volatile boolean QS = false; 13. static boolean serialVisual = true; 14. void setup(){ 15. pinMode(blinkPin,OUTPUT); 16. pinMode(fadePin,OUTPUT); 17. Serial.begin(9600); 18. BTSerial.begin(9600); 19. Bluetooth.begin(9600); 20. interruptSetup(); 21. } 22. void loop(){ 23. serialOutput() ; 24. if (QS == true){ 25. fadeRate = 255; 26. serialOutputWhenBeatHappens(); 27. QS = false; 28. } 29. ledFadeToBeat(); 30. delay(1000); 31. } 32. void ledFadeToBeat(){ 33. fadeRate -= 15; 34. fadeRate = constrain(fadeRate,0,255); 35. analogWrite(fadePin,fadeRate); 36. } |

[[9]](#footnote-9)

This source code is based on open source about measuring heart rate. And we put some source code that we need.

Line 1 ~ 4 : Set up pin numbers for serial communication.

Line 22 ~ 24 : Set up bluetooth speed of communication and serial baud speed

[[10]](#footnote-10)AllSerialHandling.ino

|  |
| --- |
| 1. #define BPMCount 3 2. void serialOutput(){ 3. if (serialVisual == true){ 4. arduinoSerialMonitorVisual('-', Signal); 5. } else{ 6. sendDataToSerial('S', Signal); 7. } 8. } 9. void serialOutputWhenBeatHappens(){ 10. if (serialVisual == true){ 11. Serial.print("\*\*\* Heart-Beat Happened \*\*\* "); 12. Serial.print("BPM: "); 13. Serial.print(BPM); 14. Serial.print(" "); 15. static int BPMS[BPMCount] = { 0 }; 16. static int count = 0; 17. if(BPM <= 160 && BPM >= 40 ){ 18. if(count == 0){ 19. BPMS[count++] = BPM; 20. }else if(abs(BPMS[count-1] - BPM) < 9 && count < BPMCount){ 21. BPMS[count++] = BPM; 22. }else{ 23. memset(BPMS,0,sizeof(int)\*BPMCount); 24. count = 0; 25. BPMS[count++] = BPM; 26. } 27. if(count - 1 == BPMCount - 1){ 28. int AvrBPM = 0; 29. count = 0; 30. for(int i=0;i<BPMCount;i++){ 31. AvrBPM += BPMS[i]; 32. Serial.print(BPMS[i]); 33. Serial.print(" "); 34. [[11]](#footnote-11) } 35. AvrBPM /= BPMCount; 36. BTSerial.write((char)(AvrBPM-40)); 37. delay(1); 38. Serial.print(AvrBPM); 39. memset(BPMS,0,sizeof(int)\*BPMCount); 40. count = 0; 41. } 42. } 43. } else{ 44. sendDataToSerial('B',BPM); 45. sendDataToSerial('Q',IBI); 46. } 47. } 48. void sendDataToSerial(char symbol, int data ){ 49. Serial.print(symbol); 50. Serial.println(data); 51. } 52. void arduinoSerialMonitorVisual(char symbol, int data ){ 53. const int sensorMin = 0; 54. const int sensorMax = 1024; 55. int sensorReading = data; 56. int range = map(sensorReading, sensorMin, sensorMax, 0, 11); 57. switch (range) { 58. case 0: 59. Serial.println(""); 60. break; 61. case 1: 62. Serial.println("---"); 63. break; 64. case 2: 65. Serial.println("------"); 66. [[12]](#footnote-12) break; 67. case 3: 68. Serial.println("---------"); 69. break; 70. case 4: 71. Serial.println("------------"); 72. break; 73. case 5: 74. Serial.println("--------------|-"); 75. break; 76. case 6: 77. Serial.println("--------------|---"); 78. break; 79. case 7: 80. Serial.println("--------------|-------"); 81. break; 82. case 8: 83. Serial.println("--------------|----------"); 84. break; 85. case 9: 86. Serial.println("--------------|----------------"); 87. break; 88. case 10: 89. Serial.println("--------------|-------------------"); 90. break; 91. case 11: 92. Serial.println("--------------|-----------------------"); 93. break; 94. } 95. } |

Line 1 : Macro variance, the number of times to extract the right BPM.

Line 17 : Static array to store the correct BPM

Line 18 : A static variable that counts the measured BPM

Line 19 ~ 28 : If the measured BPM value is between 40 and 160 and there is no BPM value in the first array, it is stored in it. When the BPM value is stored in the array, it is stored in the array when the absolute value of the difference is less than 9 as compared with the previous BPM, and if not, the array is initialized and the BPM counting is initialized.

Line 29 ~ 42 : If the BPM values are extracted as many as the number of extracted BPMs, the BPM values are averaged and the values are sent to the Arduino via serial communication and the array is initialized.

[[13]](#footnote-13)

**Arduino UNO Source Code**

|  |
| --- |
| 1. #include<SoftwareSerial.h> 2. #define Red 9 3. #define Red2 11 4. #define Green 5 5. #define Green2 10 6. #define Blue 6 7. #define Blue2 3 8. SoftwareSerial BTSerial(8,7); 9. int LED[2][3] = {Red,Green,Blue,Red2,Green2,Blue2}; 10. void setup() 11. { 12. for(int i=0;i<2;i++){ 13. for(int j=0;j<3;j++){ 14. pinMode(LED[i][j], OUTPUT); 15. [[14]](#footnote-14) digitalWrite(LED[i][j],LOW); 16. } 17. } 18. Serial.begin(9600); 19. BTSerial.begin(9600); 20. } 21. void loop(){ 22. int num = 0; 23. int chk = 0; 24. int light = 0; 25. int checkColor=0; 26. int countValue = 0; 27. if(BTSerial.available()){ 28. num = BTSerial.read() + 40; 29. if(num < 150 && num > 40){ 30. Serial.println(num); 31. countValue = num \* 2; 32. while(chk++ <= countValue){ 33. while(light+=5<255){ 34. if(checkColor == 0){ 35. analogWrite(Red,light); 36. analogWrite(Red2,light); 37. }else if(checkColor == 1){ 38. analogWrite(Green,light); 39. analogWrite(Green2,light); 40. }/\*else if(checkColor == 2){ //파란색 41. analogWrite(Blue,light); 42. analogWrite(Blue2,light); 43. }\*/ 44. if(light+5>255){ 45. break; 46. } 47. delay(4.69 \* 60 / num); 48. } 49. light = 0; 51. checkColor = 1 - checkColor; 52. digitalWrite(Red,LOW); 53. digitalWrite(Red2,LOW); 54. digitalWrite(Green,LOW); 55. digitalWrite(Green2,LOW); 56. digitalWrite(Blue,LOW); 57. digitalWrite(Blue2,LOW); 58. } 59. } 60. } 61. } |

Line 10 : Declared to be serial communication at Jarduino-BT-mini

Line 12 : Put the pin numbers of the LEDs into the two-dimensional array.

Line 37 : A loop that blinks the LEDs until the song is nearly finished.

Line 53 : In order to blink the LEDs similar to the BPM speed, use delay.

Line 57 : It is the part where Red-LEDs flashes and then Green-LEDs flashes.

[[15]](#footnote-15)

I’ll talk about the raspberry pi coding. Raspberry pi played a song according to its BPM. It can receive BPM value through heart rate sensor.

스크린샷이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명스크린샷이(가) 표시된 사진

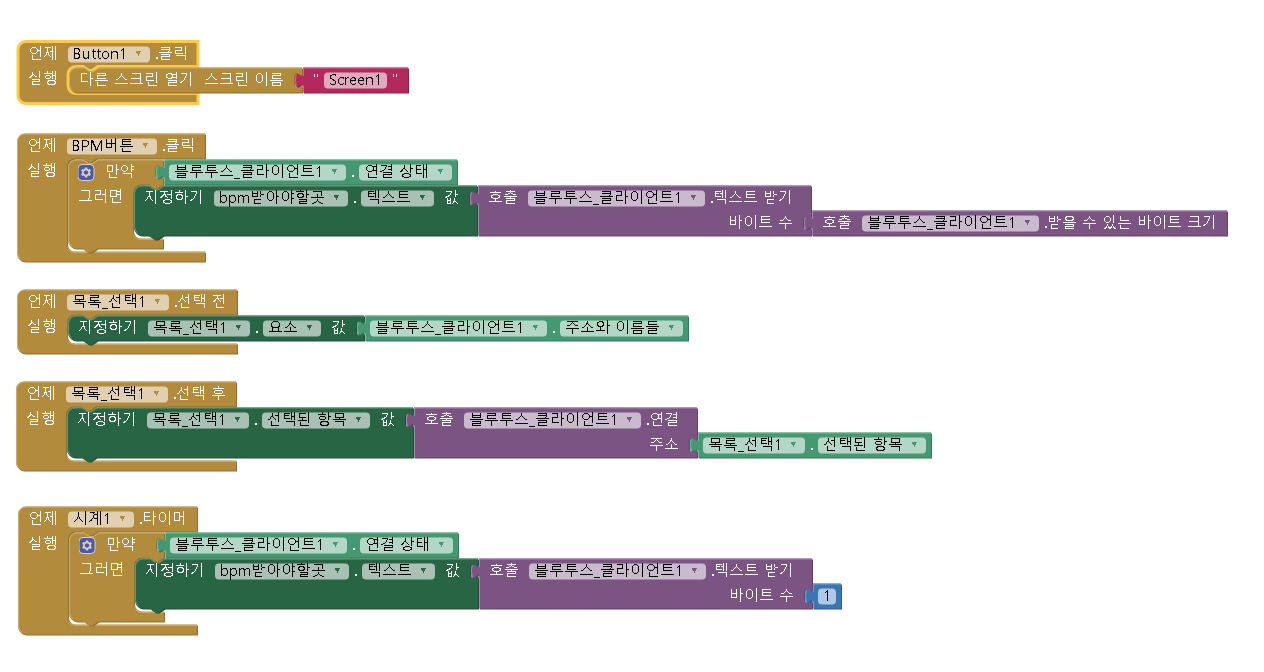
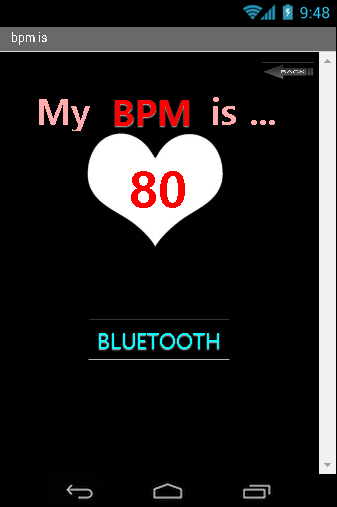
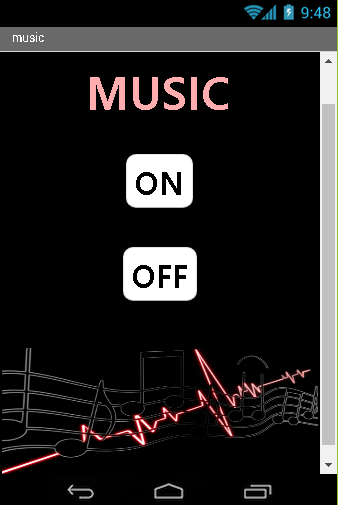
높은 신뢰도로 생성된 설명

[[16]](#footnote-16)This is a coding of raspberry pie. If you look at the while statement, you can see that you have coded to play the right song for each BPM section. In Python, Arduino prints the BPM value on the serial screen using serial communication. At this point, RaspberryPi reads this. We used simple os system programming to integerize the BPM value read and to play mp3 in the Linux kernel. In the Raspberry Python kernel, you can play MP3s in various ways, which you play using a package called omxplayer. I will explain this coding in more detail.

|  |
| --- |
| #!/usr/bin/env python --> Part for automatic execution  import os --> The system function, which is the file to load in order to run commands in the Linux kernel on RaspberryPi.  import serial --> A file that is read from the Arduino to read the value that is on the serial monitor by serial communication  import time --> sleep, The file to be loaded to use the time delay.  import random --> File to fetch Random value.  list = ['0','1','2']  --> Arrangement for finding the USB port connected to Arduino (connect to Arduino, one of ttyUSB0 ~ ttyUSB2 or ttyACM0 ~ ttyACM2)  portcount = 0 --> The variable that specifies the position in the list array.  port = ["/dev/ttyUSB","/dev/ttyACM"] --> Variable that stores the location where USB of Arduino is connected.  portListCheck = 0 --> If there is no ttyUSB connected to the arduino, the variable to connect to ttyACM.  --> ex) It is in the USB portListCheck = 0 and it is used as port [portListCheck] when declaring USB position later.  omxplayerMusic = 'omxplayer -o local ~/Desktop/AED/' --> Variable that tells the song location.  --> omxplayer -o local ~/Destop/AED/155\_1.mp3 ==> Play 155\_1.mp3 songs in AED file with external speakers.  while(os.system('ls /dev/ttyUSB'+list[portcount]) != 0): --> Until USB connected to Arduino is confirmed.  portcount = portcount + 1 --> It values ​​in the following list.  if(portcount >= 3):  portListCheck = 1 --> Variable to be passed to ttyACM because it is not in ttyUSB.  portcount = 0 --> The first time you go back to the list array.  break --> Exiting the while.  [[17]](#footnote-17)  omxplayMusic = 'omxplayer -o local ~/Desktop/AED/' --> The variable that stores a portion of the command to play the song later.  while(os.system('ls /dev/ttyACM'+list[portcount]) != 0): --> Repeat until ttyACM is confirmed to be connected to the arduino.  portcount = portcount + 1 --> Variables for multiplying '0', '1', '2' in the list array + 1  if(portcount >= 3): --> If you have run through the list array,  break --> Exiting the while.  serialFromArduino = serial.Serial(port[portListCheck]+list[portcount],9600); --> Arduino serial communication initialization, if it is not connected to Arduino, error termination and autorun termination and raspberry pie execution.  serialFromArduino.flushInput() --> Clear Input Buffer.  MusicNum = 0 --> Variables to assign an average value within the range of songs to play.  musicrand = 0 --> Random playback using random function when there are multiple songs.  while True: --> Infinite repeat  input\_s = serialFromArduino.readline() --> Function to read the value of serial monitor.  input = int(input\_s) --> The process of converting a value read from a serial monitor to an integer.  print(input) --> Output measured BPM values ​​to Linux kernel.  [[18]](#footnote-18)  if(input >= 40 and input <= 155): --> When the value of BPM is the correct value (since the BPM value is filtered before this, the correct value fits in the if statement, but sometimes it gives an error, so it specifies the range)  if(input >= 150):  [[19]](#footnote-19)  MusicNum = 155  musicrand = int(random.random()) % 2 + 1  os.system(omxplayerMusic + str(MusicNum)+'\_'+str(musicrand)+'.mp3') --> omxplayerMusic + str(MusicNum) + '\_' + str(musicrand) + '.mp3' ==> omxplayer -o local ~/Desktop/AED/155\_1.mp3 or omxplayer -o local ~/Desktop/AED/155\_2.mp3  elif(input >= 140):  MusicNum = 145  musicrand = int(random.random()) % 2 + 1  os.system(omxplayerMusic + str(MusicNum)+'\_'+str(musicrand)+'.mp3')  elif(input >= 130):  MusicNum = 135  os.system(omxplayerMusic + str(MusicNum)+'\_1.mp3')  elif(input >= 110):  MusicNum = 125  os.system(omxplayerMusic + str(MusicNum)+'\_1.mp3')  elif(input >= 100):  MusicNum = 105  os.system(omxplayerMusic + str(MusicNum)+'\_1.mp3')  elif(input >= 90):  MusicNum = 95[[20]](#footnote-20)  musicrand = int(random.random()) % 2 + 1  os.system(omxplayerMusic + str(MusicNum)+'\_'+str(musicrand)+'.mp3')  elif(input>=80):  MusicNum = 85  musicrand = int(random.random()) % 2 + 1  os.system(omxplayerMusic + str(MusicNum)+'\_'+str(musicrand)+'.mp3')  elif(input>=70):  MusicNum = 75  musicrand = int(random.random()) % 2 + 1  os.system(omxplayerMusic + str(MusicNum)+'\_'+str(musicrand)+'.mp3')  elif(input>=60):  MusicNum = 65  musicrand = int(random.random()) % 2 + 1  os.system(omxplayerMusic + str(MusicNum)+'\_'+str(musicrand)+'.mp3')  elif(input>=50):  MusicNum = 55  musicrand = int(random.random()) % 2 + 1  os.system(omxplayerMusic + str(MusicNum)+'\_'+str(musicrand)+'.mp3')  elif(input>=40):  MusicNum = 45  os.system(omxplayerMusic + str(MusicNum)+'\_1.mp3')  else:  print("ERROR BPM") |

[[21]](#footnote-21)

Here's how we created the app: At first, I planned to make an app and use Bluetooth to get information about my heart rate in my app. We created an app using App Inventor. We can receive and confirm the measured BPM value using Bluetooth. Moreover, Music on/off can be controlled. This tells us the average heart rate value, so we can compare to our own.

   스크린샷이(가) 표시된 사진

높은 신뢰도로 생성된 설명

<Figure 8 – App inventor>

Finally, I’ll talk about the expected effect. As we prepare the project, we visually check our status through the mood, increase accessibility to the bit sign, combine with other fields through the convergence of BT and IT, We prepared the project in anticipation of the future that can help to stabilize the mind while listening to the music of BPM.

A video about our production and demonstration has been uploaded to YouTube. ( <https://youtu.be/gLO9BRZIaoA> : This is the URL )[[22]](#footnote-22)

1. Written By Hye-jeong Kim [↑](#footnote-ref-1)
2. Written By Hye-jeong Kim [↑](#footnote-ref-2)
3. Written By Ye-ram Lee [↑](#footnote-ref-3)
4. Written By Ye-ram Lee [↑](#footnote-ref-4)
5. Written By Jae-hwan Kim [↑](#footnote-ref-5)
6. Written By Jae-hwan Kim [↑](#footnote-ref-6)
7. Written By Min-SKim [↑](#footnote-ref-7)
8. Written By Min-Sang Kim [↑](#footnote-ref-8)
9. Written By Min-Sang Kim [↑](#footnote-ref-9)
10. Written By Min-Sang Kim [↑](#footnote-ref-10)
11. Written By Min-Sang Kim [↑](#footnote-ref-11)
12. Written By Min-Sang Kim [↑](#footnote-ref-12)
13. Written By Min-Sang Kim [↑](#footnote-ref-13)
14. Written By Min-Sang Kim [↑](#footnote-ref-14)
15. Written By Min-Sang Kim [↑](#footnote-ref-15)
16. Written By Ji-Eun Lee [↑](#footnote-ref-16)
17. Written By Ji-Eun Lee [↑](#footnote-ref-17)
18. Written By Ji-Eun Lee [↑](#footnote-ref-18)
19. Written By Ji-Eun Lee [↑](#footnote-ref-19)
20. Written By Ji-Eun Lee [↑](#footnote-ref-20)
21. Written By Ji-Eun Lee [↑](#footnote-ref-21)
22. Written by Ji-eun Jeong [↑](#footnote-ref-22)