HEAR THE NUMBERS

ARDUINO PROTOTYPING PROJECT

Vamsi Pasupuleti
Sijie Yang

CONTENTS

INTRODUCTION

Watch the demo video of our interactive dice
http://vimeo.com/78231959

In this project, we aimed at a marginalized group of people whose eyesights are too defective to see details on even very close objects. The intention of our design is enabling those people to play a dice in a conventional as well as familiar way, but “feel” the numbers intuitively on each side of it through interactive technology, which is Arduino platform in our case. With this idea in mind, we finally made an interactive fuzzy dice by using one Arduino board, one RobotGeek sensor shield, one buzzer, six tilt sensors and several wires. The prototype has been programmed to play six distinguishable sound patterns that represent figures from 1 to 6 respectively when it becomes stabilized on any flat surfaces (e.g. table, floor) after people throw it out. Thus people who are not able to see it can easily tell which number is facing up by hearing and comprehending meaningful sound patterns. Each pattern can be played repetitively for confirmation purpose.

RESEARCH

Concepting Phase

Initially we started with online research which was inspired by our discussion about creating not only playful toys with interactive technology but also meaningful ones that trigger people's reflection upon conventional norms. In other words, we would like to induce people to rethink about existing ways of interaction through exploration and redesign of a toy which has socially or culturally habitual forms. This intention led us to look into things such as traditional Indian toys. In order to broaden our view and obtain deeper understandings about current trend of toy making as a comparison, we conducted field observation at one of the local stores, Kmart (picture 1). On-site research gave us the chance that more potential can be identified by going through large amounts of available toys. Most of them are basically representations of roles and activities that are characterized by playfulness. For examples, apart from variant sizes, colors and materials, the major difference among toys cars is seemingly their emulational or fantastic appearance. However, there are significantly multiple ways of playing with them. Such variance in interaction became an interesting entry point where we finally found our opportunity during that field research. When we randomly grabbed a package of dices from one of those shelves, we realized that dice is a conventional toy which has existed for plenty of time but its possibilities of being interactive hasn't been well reconsidered, especially under the circumstance that technology enables more and more opportunities to be discovered (picture 2). Thus we came to the agreement that we wanted to implement an interactive dice by taking advantage of Arduino development kit in a way that benefits its users, namely dice players whom we defined as the blind in general, including people whose eyesights are impaired temporarily. Detailed rationale of selecting our

RESEARCH

target user group can be found in the following Design section.

Picture 1. On-site Observation at Kmart

Picture 2. A Package of Dice at Kmart

RESEARCH

Implementing Phase

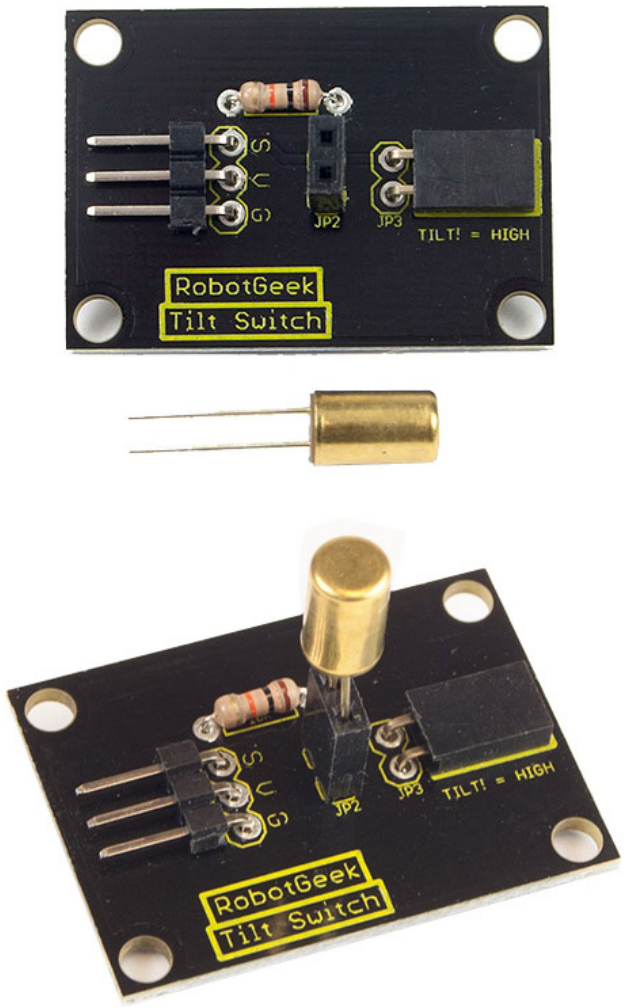
After we decided to make a dice for people who are unable to see things while playing with it, acoustic and haptic feedbacks are two basic options that can be used for complementing the lack of visual information. As a result, the idea of making sounds or vibrations or both came into our minds. Our biggest challenge here was to figure out ways of producing sounds or vibrations which represent the numbers on each side of the dice appropriately. Then parallel research was conducted so that we could find out what kind of Arduino compatible sensors may be useful for either sound or vibration generating purpose, as well as some sort of mechanism that allows the sensors to respond to dice's six dimensions according to its landing position.

We found out online that tilt sensors (picture 3) which connect or discount the circuit based on its positioning and the influence of gravity can be an affordable solution. The reason why we resorted to those new sensors as opposed to only using other sensors we had at hand was that what we got from the development kit didn't meet our needs both in terms of quantity and functionality. We would otherwise have made the most use of sensors from the kit if a particular sensor was not limited to its own particular usage. In fact, such constraint led us to start thinking about taking advantage of tilt sensors. Our research in this phase unfolded the prototyping idea that we can represent each figure by placing six tilt sensors on each side of a dice. No matter which side ends up facing up, the specific tilt sensor which stand for that side will trigger sounds to be played through the buzzer, or vibrations to be produced through vibration motor. However, there is no compatible vibration motor that is specialized for Arduino platform so far. Hence, we moved on by focusing on providing acoustic feedback only.

RESEARCH

Apart from tilt sensors, we also found a sensor shield that is capable of accommodating individual sensors and Arduino board at the same time. It gave us a great opportunity to combine the shield and the board together so that they became compact enough to be put into a 3 inch x 3 inch fuzzy toy dice.

Early experiments were proceeded as long as we received those parts we ordered. Tilt sensors were very sensitive and they worked well as expected. More trials and errors appeared along way when it came to subsequent production in which we tried to make discrete things work along with each other. More details will be talked about in the remaining sections.



Picture 3. Robotgeek Tilt Sensor

DESIGN RATIONALE

Intended User Group

Our intended user group consists of people who are not able to see the numbers on the dice while they are playing with the it, including the blind population, people whose eyes are covered for gaming reasons, and users who play in the low light environments or even in darkness.
Normally a dice is supposed to be seen so that people can tell what figure is displayed at the top after being casted. But we thought there was an opportunity that we can modify the dice and make it equipped with unintended interactive capacities in order to fulfill marginalized needs that have been ignored by the public. Our prototype may not be a perfect demonstration of our intention, but we hope it serves as a modest spur to induce more people to come forward with this valuable direction.

Intended Context of Use

We want to point out an important assumption we hold here. That is, everyone deserves to be able to experience playing with any toys in a mutually meaningful way. By mutually meaningful way, we mean effective interactions and proper feedback that occur between a player and a playful object. That being said, the intended context of use can be any situations in which people who have visual impairment play games that involve dice individually or collaboratively. It also can be any circumstances in which people intentionally cover their eyes or stay in darkness while they are having fun. In addition, one interesting scenario might be using our design to test or reinforce people's mental ability of recognizing different sound patterns for physiological study and so on.

Goals for the Experience

Like we mentioned above, we want to enable people who are lack of visual sense to play a dice like how normal people do, and to be capable of

DESIGN RATIONALE

differentiating figures from 1 to 6 accurately, easily and intuitively. Our way of achieving this goal is to embed a lightweight and programmable sound making system into an ordinary dice.

Why Should HCI Care

By making this dice which has been transformed from its traditional type into a digitally interactive kind, we propose that it is worthwhile to introduce HCI into the exploration and redesign process of conventional beings.

The development of HCI in practice is often driven by curiosity in new ways of interaction. It happens not just inside people's minds but within a large picture, namely social, cultural, and historical contexts. However, if we tend to look for purely novel design space and have the general notion of user in mind, we already lost empathy on people who may use the design in real life situations. Instead of keeping our eyes on seeking innovative approaches, rethinking about existing norms, artifacts and activities that people are familiar with can help us stay on track of designing for actual users and scenarios. There are a lot of underlying opportunities to be taken. Besides, HCI should pay enough attention to marginalized population who don't have that much access to interactive technology, because we believe that interaction design should be substantially regarded as a service which is intended to empower people and improve our quality of life in a pervasive manner.

PROTOTYPING

Prototyping Process

We started with analyzing various sensors to achieve the desired outcome. Our initial idea is to use flex and accelerometer sensor to detect the touch and orientation of the dice. Upon exploring and discussing with Austin Toombs, we zeroed on using tilt sensors.

Tilt sensors can primarily classified into two types, switch based and proportional tilt sensors.
a) Switch based tilt sensors: As the name suggests, these are switch sensors just pose a question whether they are tilted or not. The output is two types: HIGH or LOW (On or Off). These are further divided into two types. Mercury switch sensor and Ball in a Cage Structure Switches. Mercury fluid is in the first type and metal ball is used for later one. The disadvantage with these types of sensor are, they just give two out types.

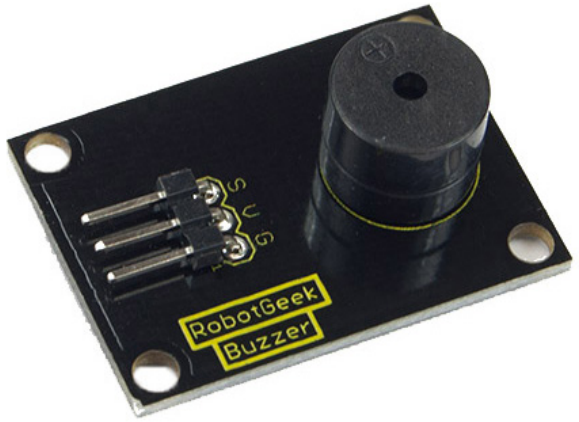
b) Proportional tilt sensors: As the name says, these sensors, the output is proportional to the degree of tilt. One can detect multiple output with these sensors. The disadvantage with these sensors are they don't long last, as they use electrolysis method to detect multiple outputs.

For our prototype, the ideal tilt sensor would be proportional tilt sensor, but due to unavailability of sensors and it short expiry period. We have chosen "Ball in a Cage Structure Switches" tilt sensors.

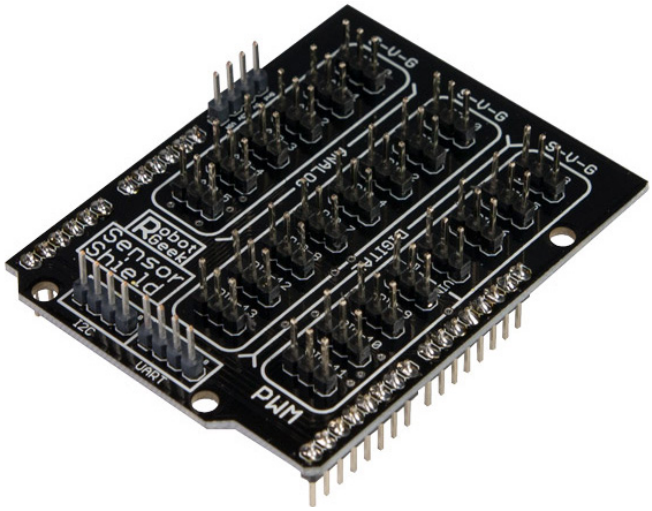
The other challenge we felt we would foresee while constructing the device is managing lot of wires (6 wires for 6 tilt sensors, one wire for buzzer, and wires joining arduino and breadboard). The thought behind 6 tilt sensors and buzzer is to place sensors on all the surface of the cube and create switch HIGH (ON) state for a specific dice facing up. Thus, every tilt switch corresponds to a

PROTOTYPING

number on the dice and based on the dice number facing up, a tilt sensor is activated and a corresponding sound played through buzzer (output). The challenging we face here is to keep arduino, breadboard and sensors inside the dice. We looked, if we can find some shields which can be attached to arduino and making it easier manage wires around the cube. On further, we discovered robotgeek manufactures making robotgeek tilt sensors, robotgeek buzzer (picture 4) and robotgeek sensor shield (picture 5).



Picture 4. Robotgeek Sensor Shield



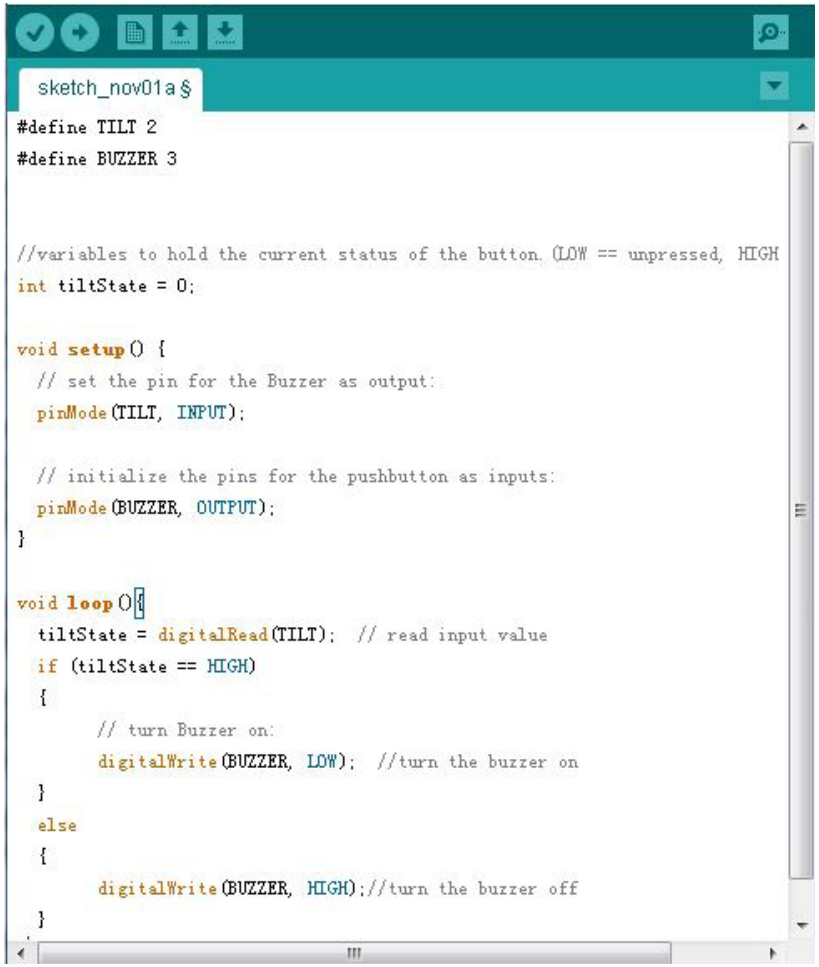
Picture 5. Robotgeek Buzzer

PROTOTYPING

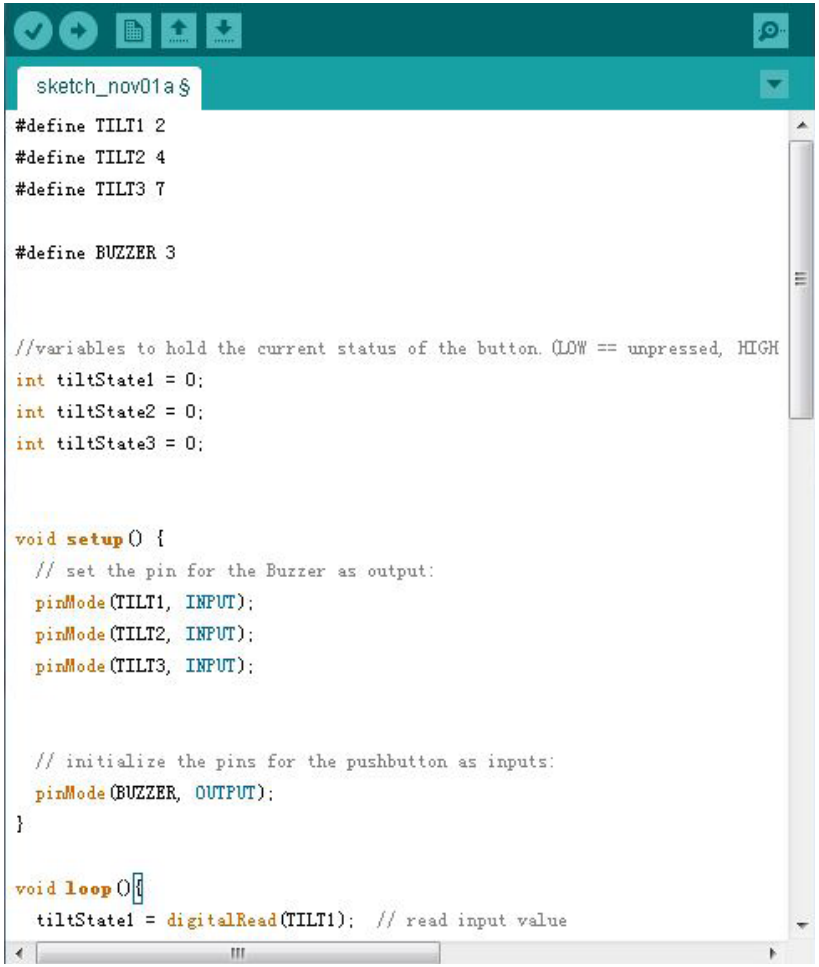
Code Overview

We started with testing the tilt sensor with changing the basic switch code (picture 6, please refer to appendix for the code).

Then we started building the code for 3 tilt sensors (picture 7, please refer to appendix for the code).



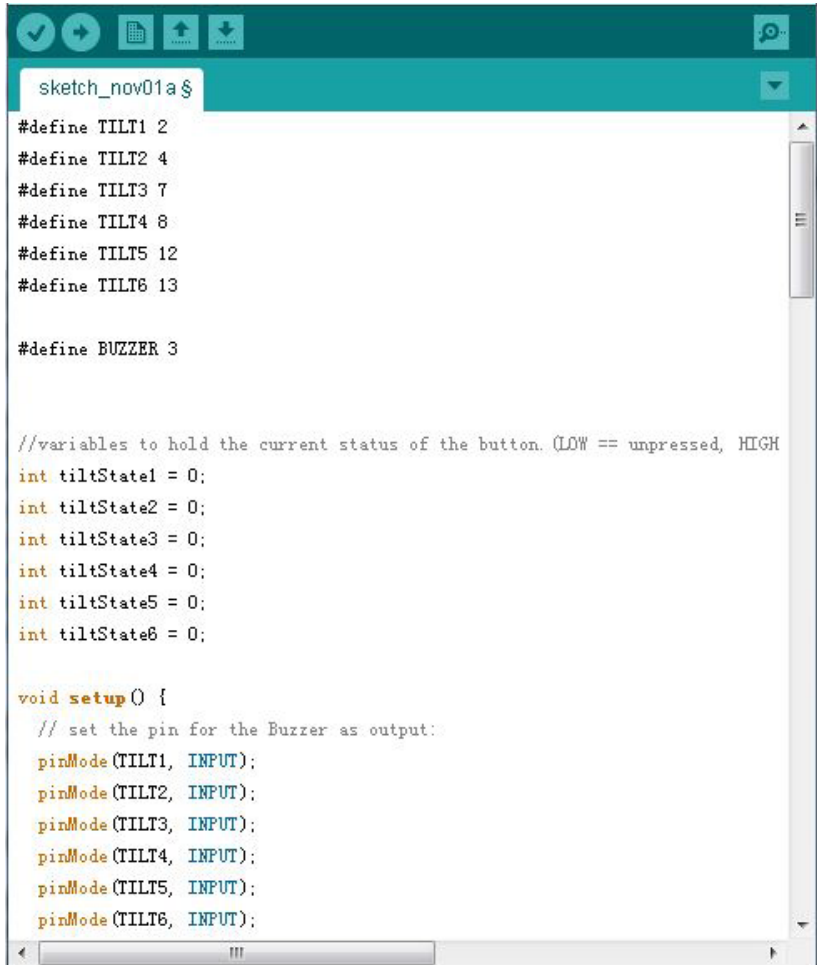
Picture 6. Basic Switch Code



Picture 7. 3 Tilt Sensors Code

PROTOTYPING

Finally, we programmed for six tilt sensors (picture 8 and picture 9, please refer to appendix for the code).



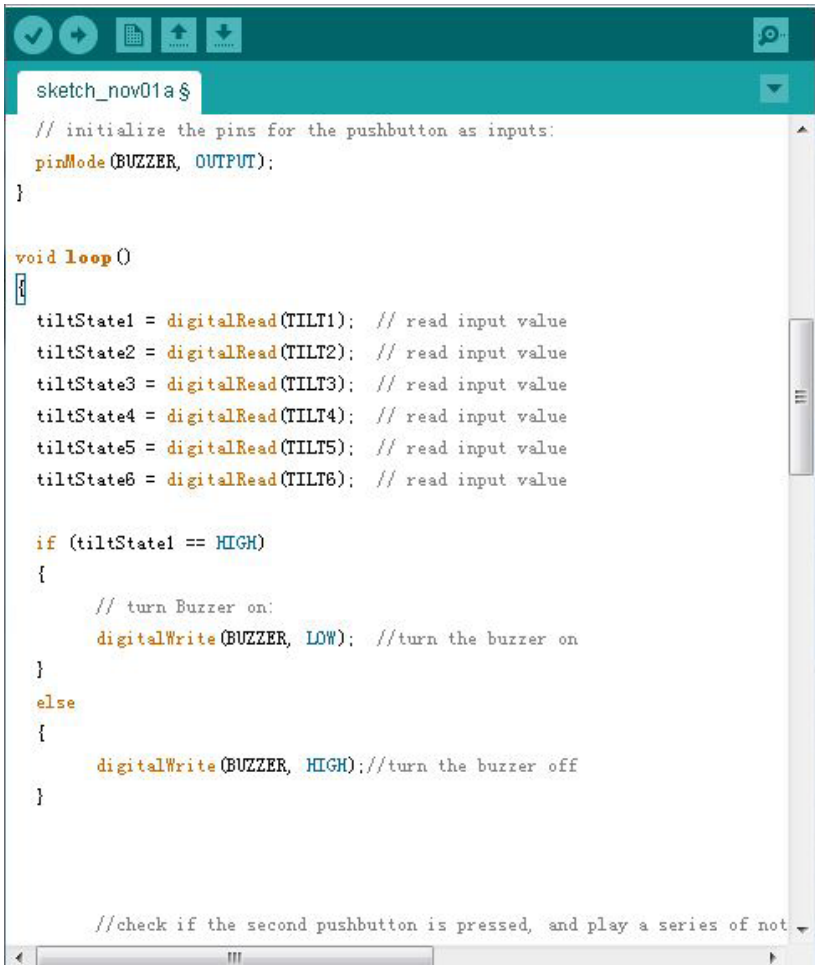
```
sketch_nov01a §
#define TILT1 2
#define TILT2 4
#define TILT3 7
#define TILT4 8
#define TILT5 12
#define TILT6 13

#define BUZZER 3


//variables to hold the current status of the button. (LOW == unpressed, HIGH
int tiltState1 = 0;
int tiltState2 = 0;
int tiltState3 = 0;
int tiltState4 = 0;
int tiltState5 = 0;
int tiltState6 = 0;

void setup() {
  // set the pin for the Buzzer as output:
  pinMode(TILT1, INPUT);
  pinMode(TILT2, INPUT);
  pinMode(TILT3, INPUT);
  pinMode(TILT4, INPUT);
  pinMode(TILT5, INPUT);
  pinMode(TILT6, INPUT);
```

Picture 8. 6 Tilt Sensors Code



```
sketch_nov01a §
  // initialize the pins for the pushbutton as inputs:
  pinMode(BUZZER, OUTPUT);
}

void loop()
{
  tiltState1 = digitalRead(TILT1);  // read input value
  tiltState2 = digitalRead(TILT2);  // read input value
  tiltState3 = digitalRead(TILT3);  // read input value
  tiltState4 = digitalRead(TILT4);  // read input value
  tiltState5 = digitalRead(TILT5);  // read input value
  tiltState6 = digitalRead(TILT6);  // read input value

  if (tiltState1 == HIGH)
  {
        // turn Buzzer on:
        digitalWrite(BUZZER, LOW);  //turn the buzzer on
  }
  else
  {
        digitalWrite(BUZZER, HIGH);//turn the buzzer off
  }



        //check if the second pushbutton is pressed, and play a series of not
```

Picture 9. 6 Tilt Sensors Code

PROTOTYPING

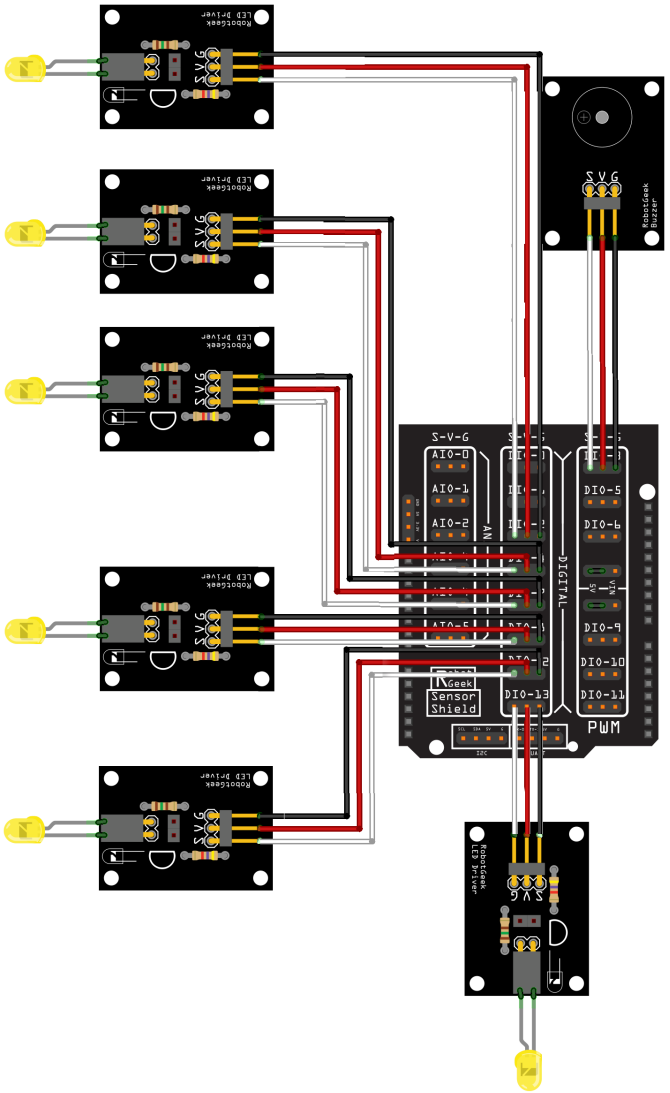
Once we felt comfortable with the code, we were looking for right dice to insert arduino, sensors and buzzer inside the dice. We found a 3 inch fuzzy dice (picuture 10), which is constructed using a form material and this is right kind of material to secure and place arduino kit.

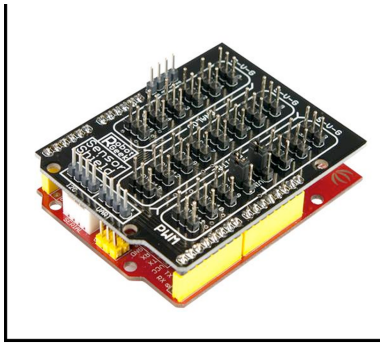
We programmed many versions of the code to find right kind of experience for the people we were designing. We were learning as we were iterating the code. Finally, we had the best suited code for all the six tilt sensors and experience (sound patterns) we intended to achieve. When we used tilt sensors on all 6 sides of the cube, we had issues with replicating the same experience as earlier. This is due to high sensitive nature of the ball moments in the tilt sensors, and we found at times there were two tilt sensors activated. To reduce the error prone situations, we are playing the sounds associated to the number on the dice in a loop (more than one time).

Picture 10. 3 Inch Fuzzy Dice

PROTOTYPING

Electronic Circuit Diagram (Robotgeek sensor shield is attached to Arduino board)

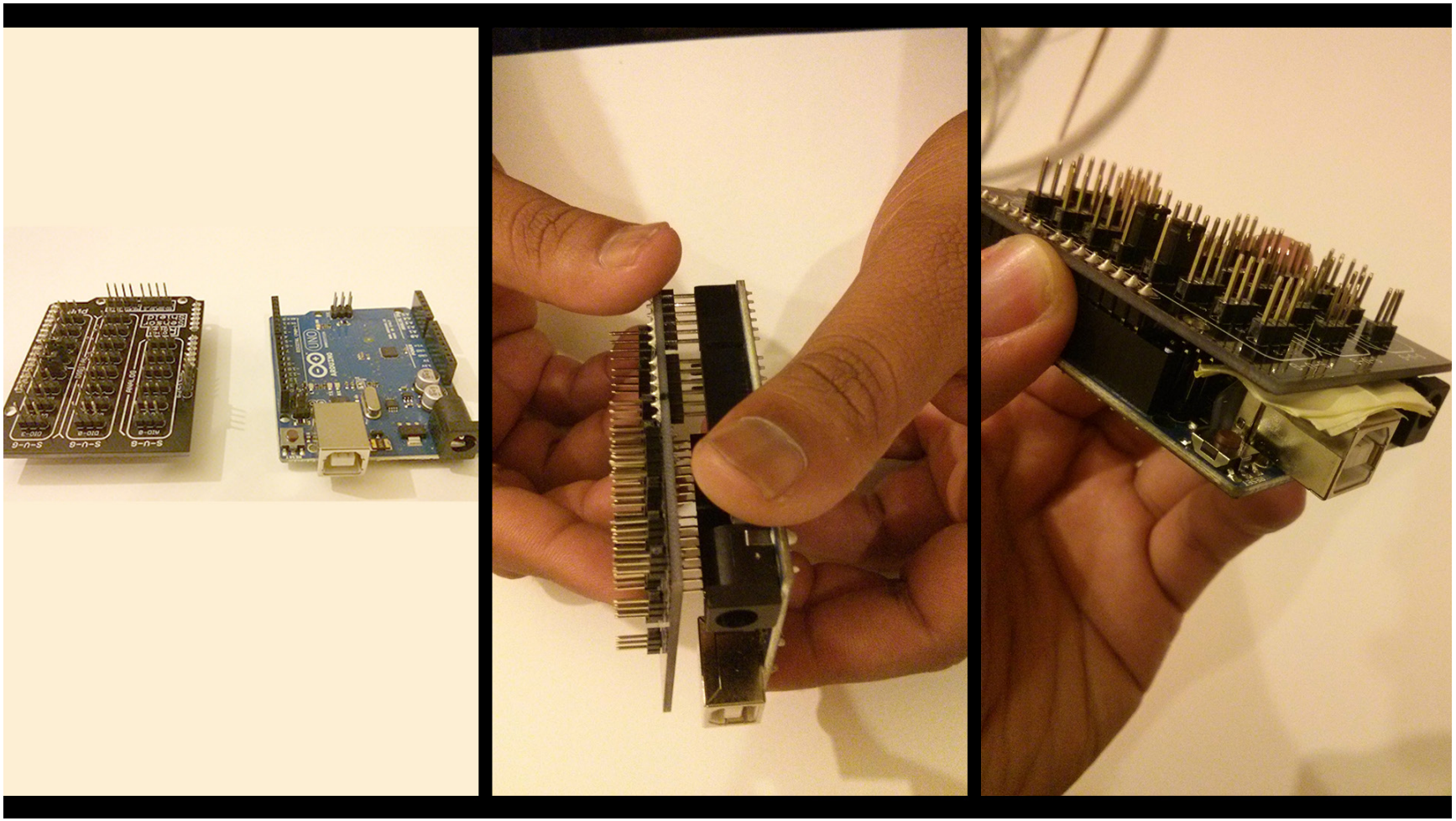




PROTOTYPING

Process Review

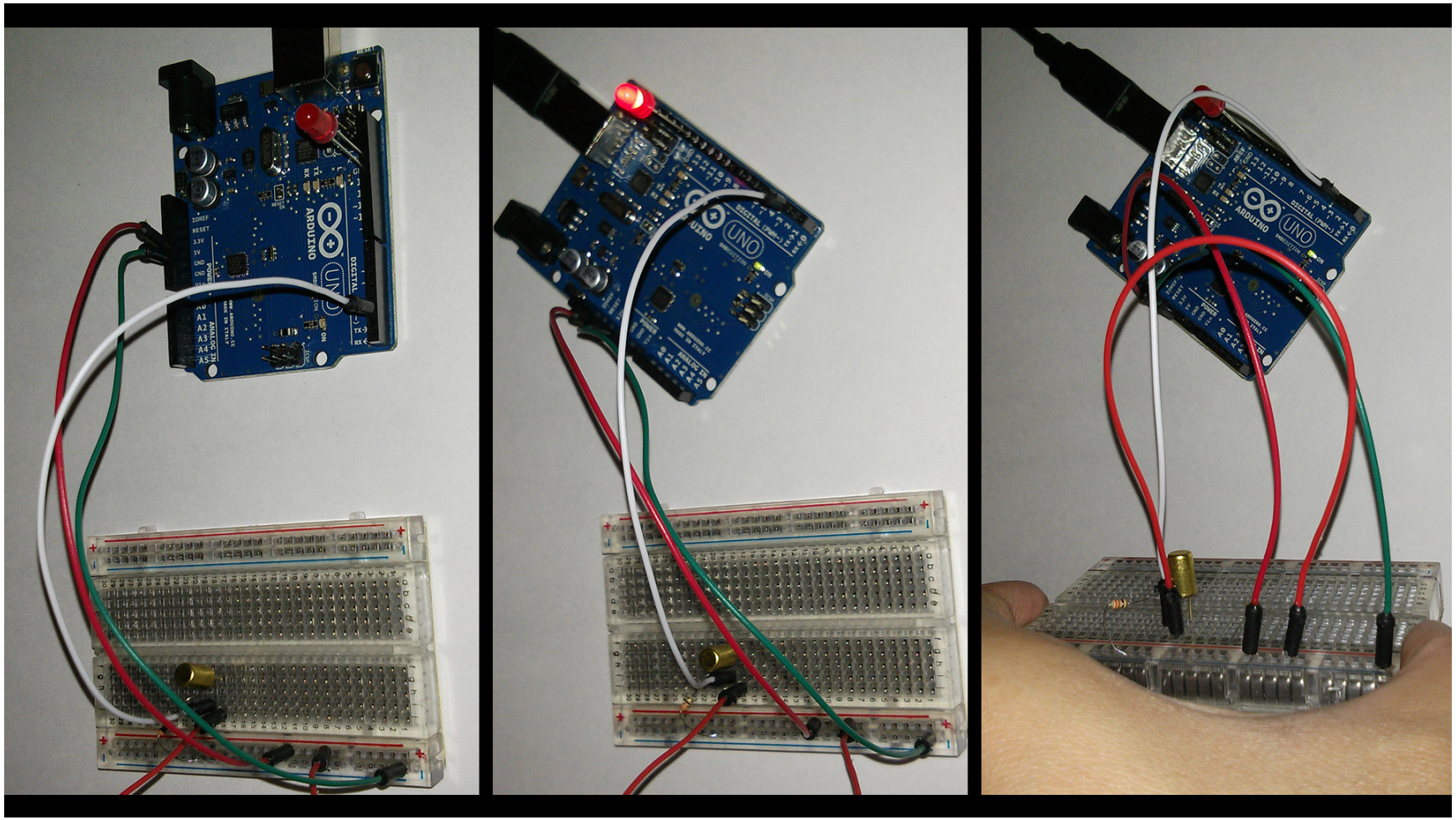
From the left to the right in picture 11, we were assembling Arduino board with Robotgeek sensor shield. A piece of paper was folded and inserted in between the USB port and the shield to avoid short circuit.



Picture 11. Assemble Arduino Board with Robotgeek Sensor Shield

PROTOTYPING

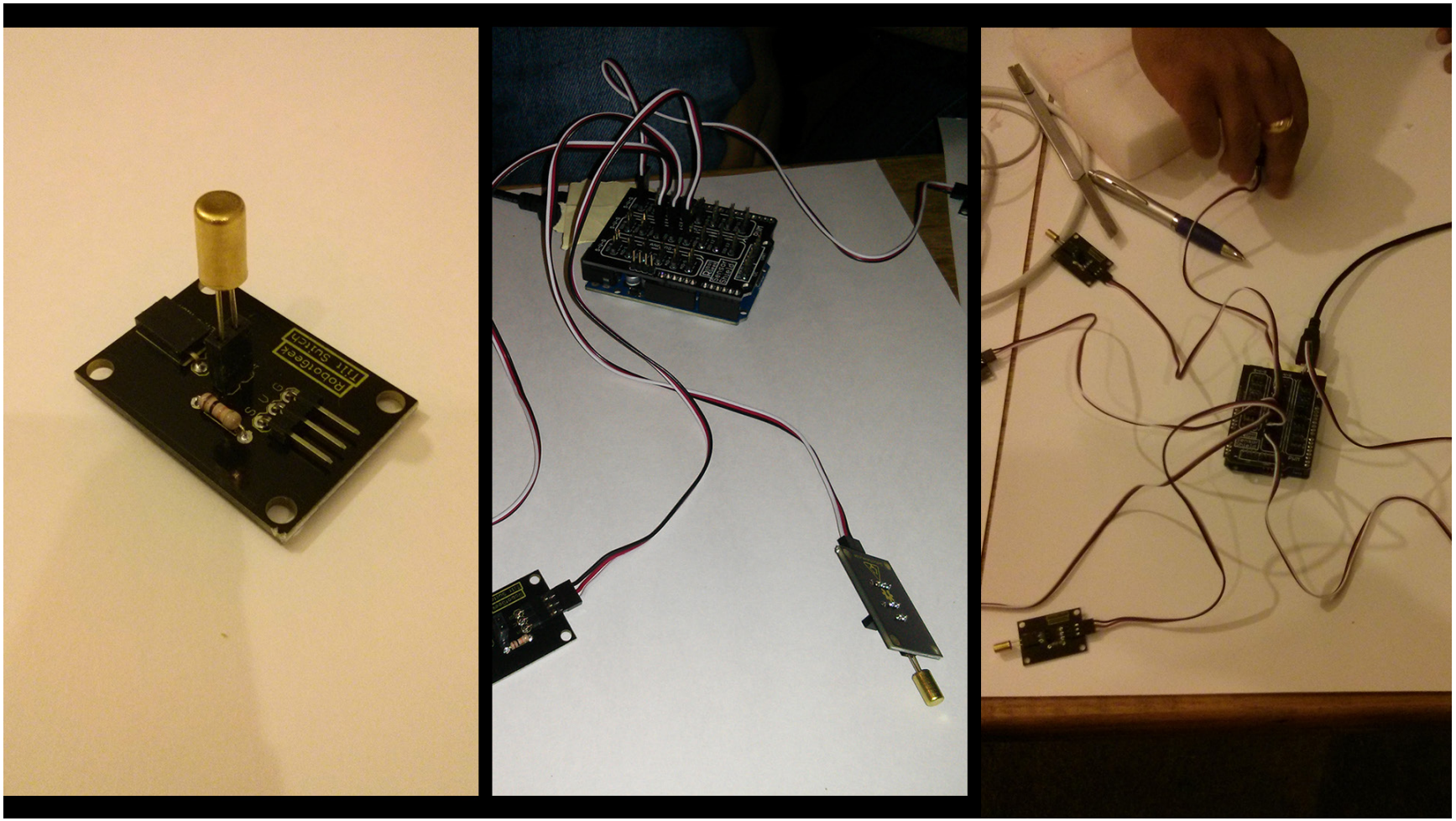
From the left to the right in picture 12, we were testing on the tilt sensor to see if it can work properly by using a LED light to provide visual feedback as confirmation.



Picture 12. Testing on Tilt Sensor

PROTOTYPING

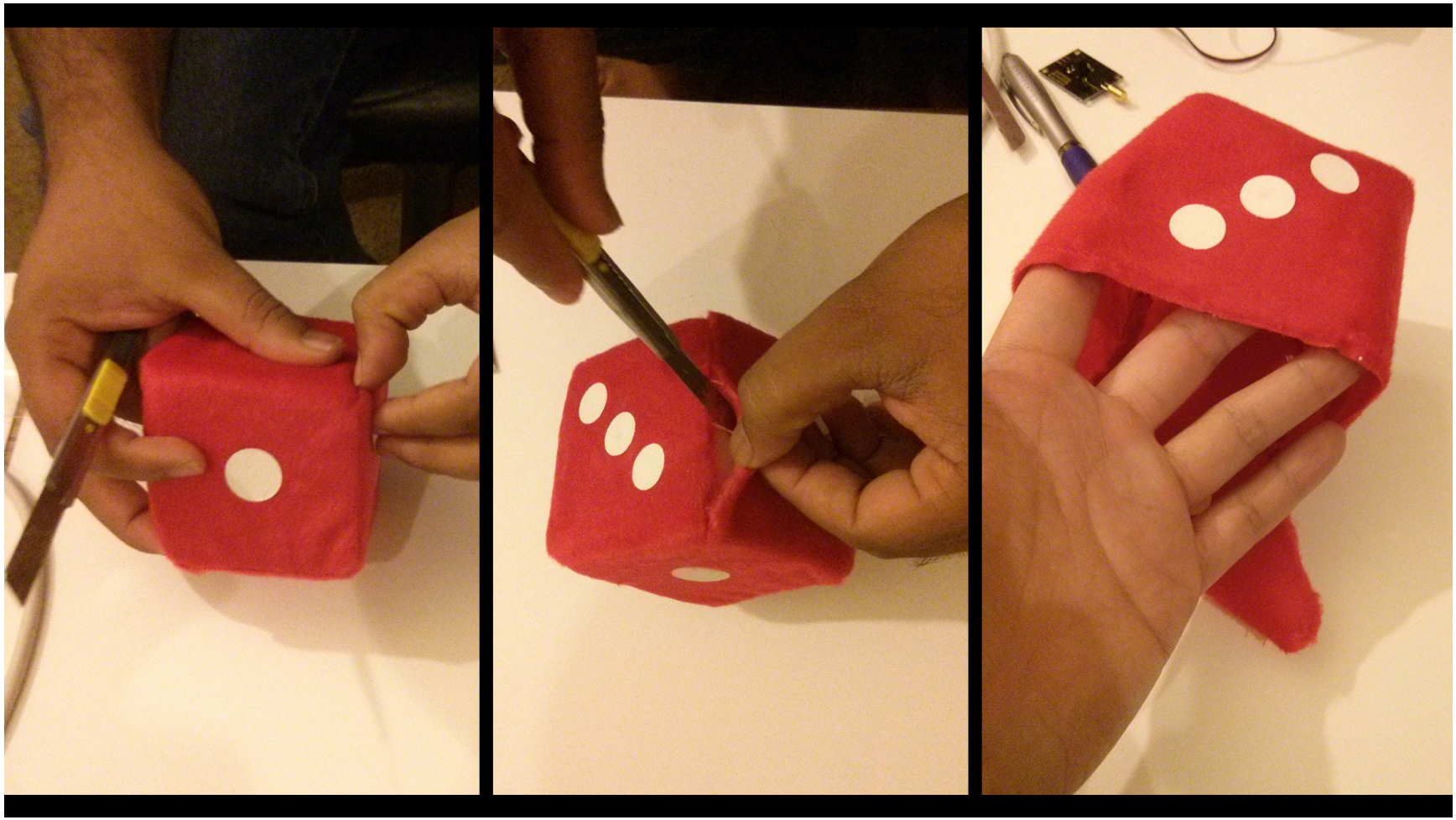
From the left to the right in picture 13, we were running tests on the situations that either three tilt senors or six sensors were combined and working together.



Picture 13. Testing on 3 and 6 Tilt Sensors Working Together

PROTOTYPING

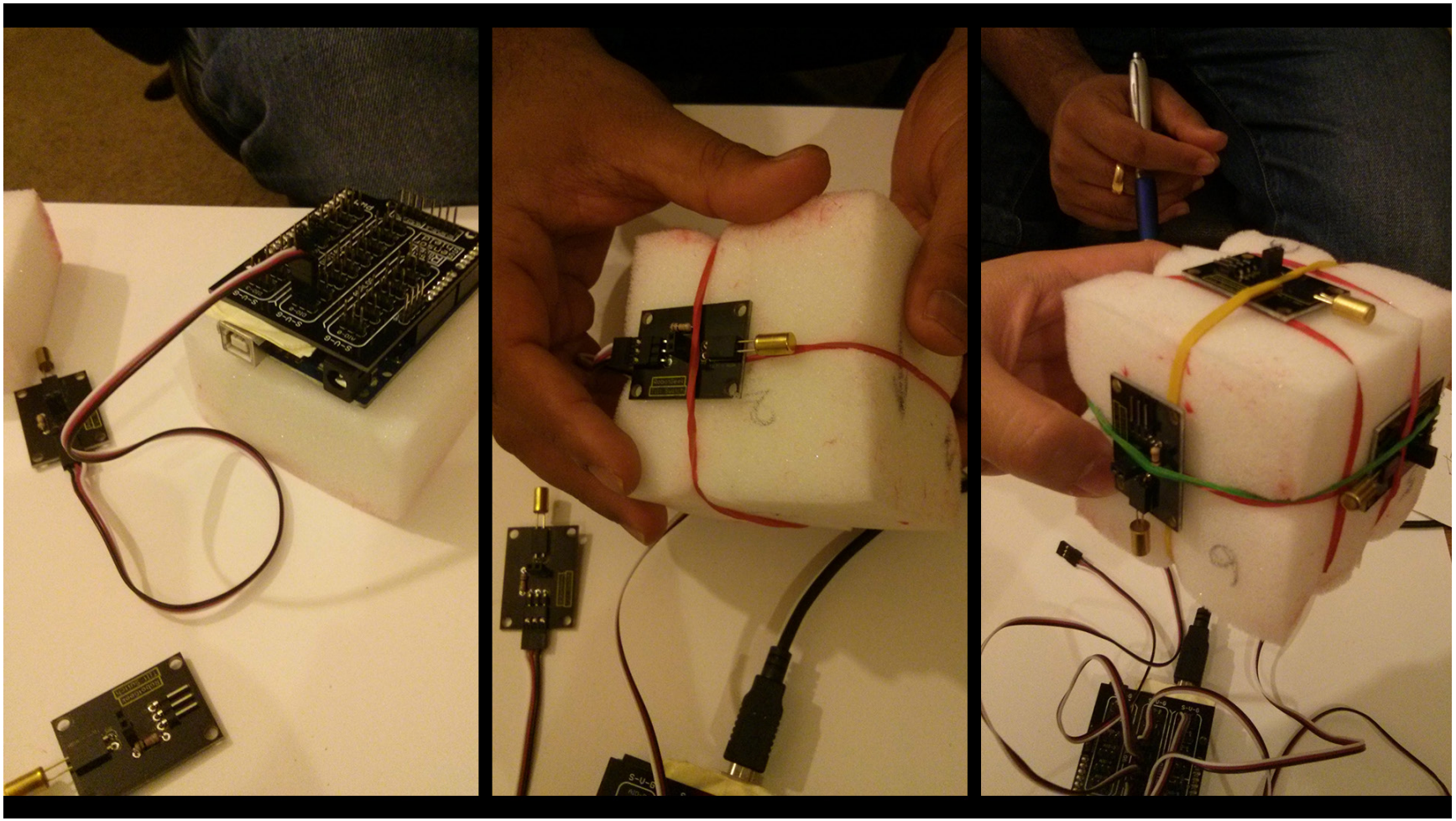
From the left to the right in picture 14, we were cutting the fuzzy toy dice and taking out its foam body. Only one side of the dice was left open so that we can put everything back into it again afterwards.



Picture 14. Cutting the Fuzzy Dice

PROTOTYPING

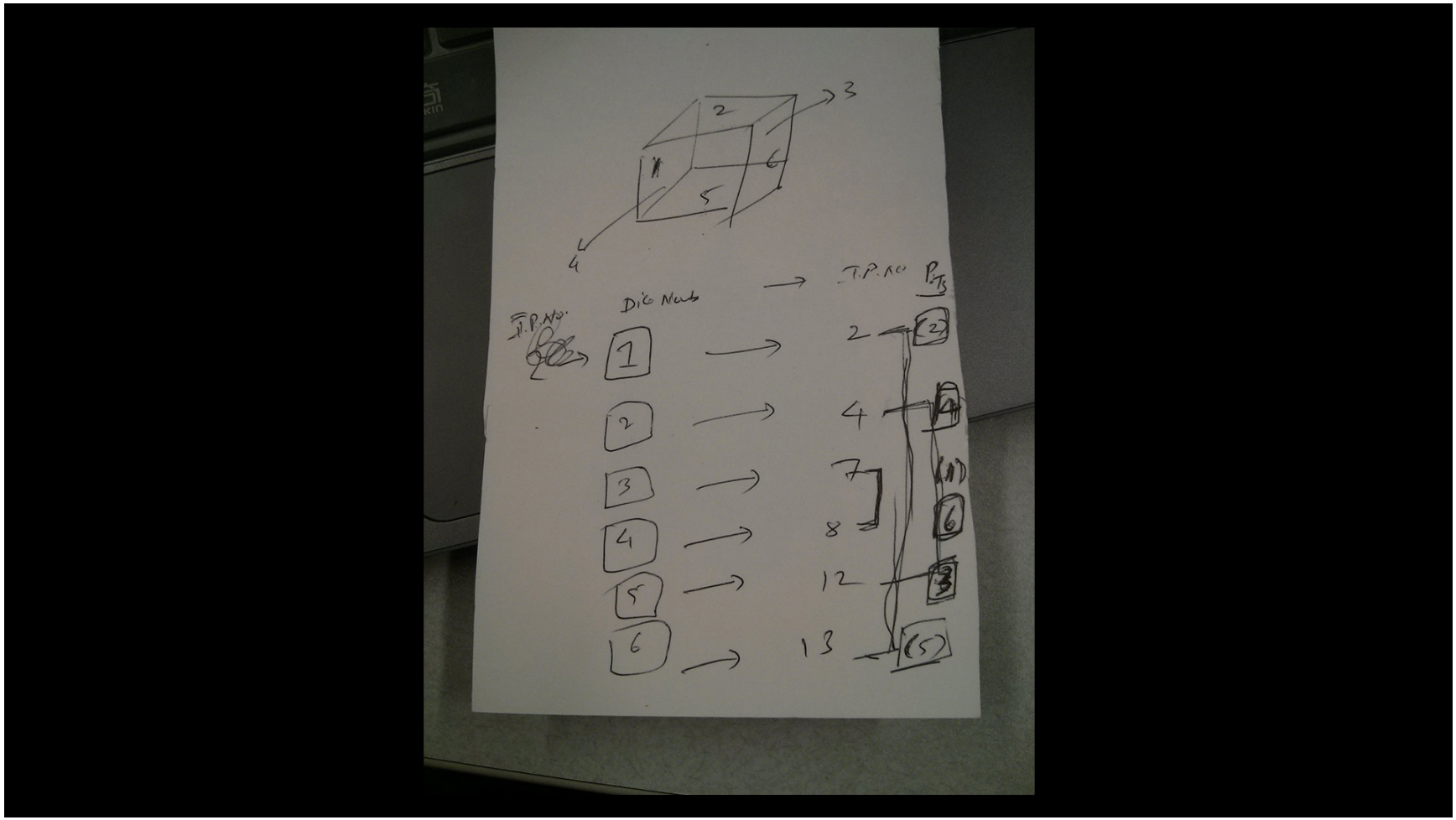
From the left to the right in picture 15, we were placing all the sensors on the surface of the foam body by using rubber bands. Each sensor was facing the side with a specific number they were representing.



Picture 15. Placing Tilt Sensors on Foam Body of the Dice

PROTOTYPING

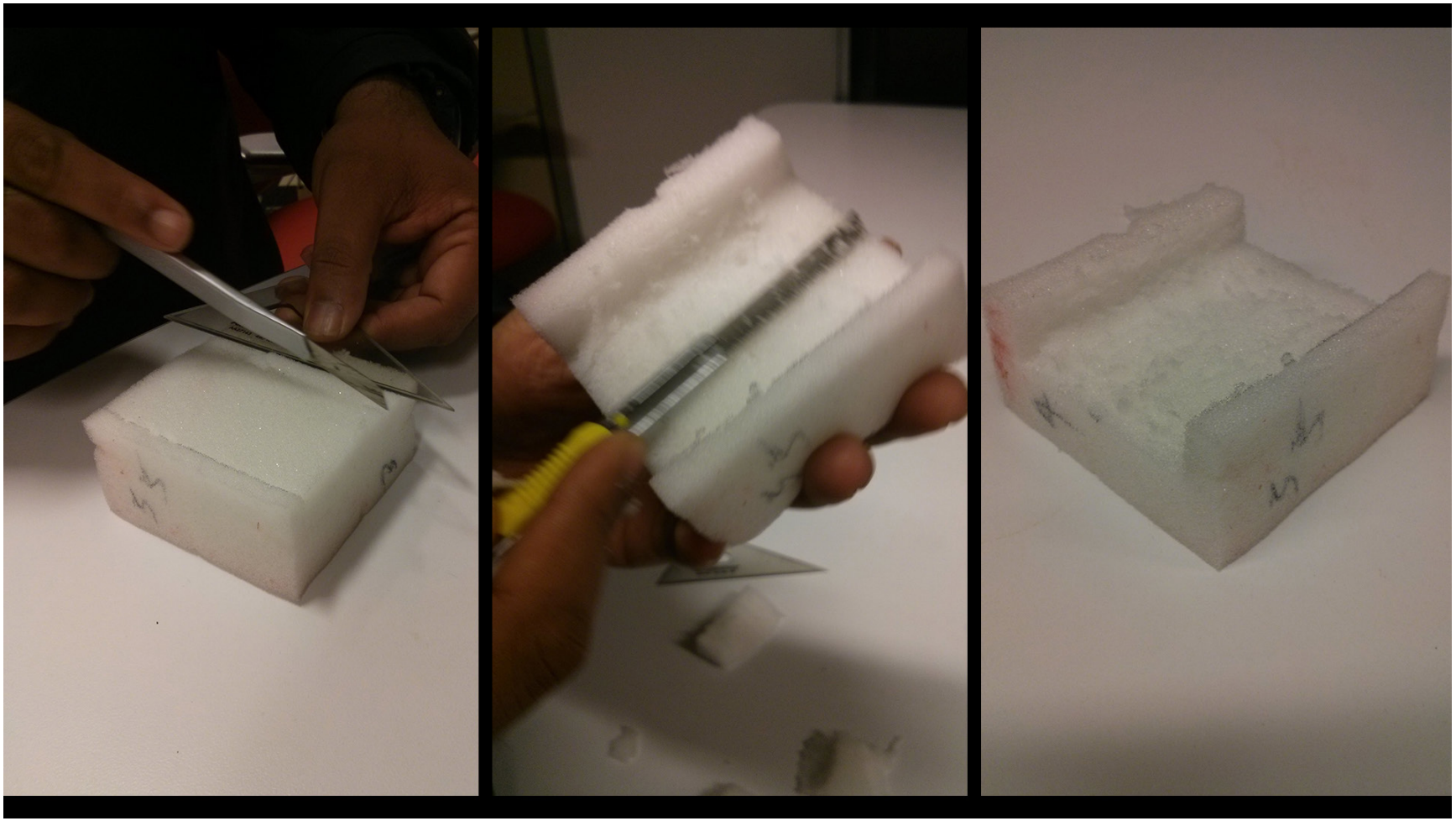
In picture 16, we were using our sketches to figure out the mapping relationships between numbers on each side of the dice and their corresponding I/O ports on the sensor shield.



Picture 16. Sketching the Mapping Relationships

PROTOTYPING

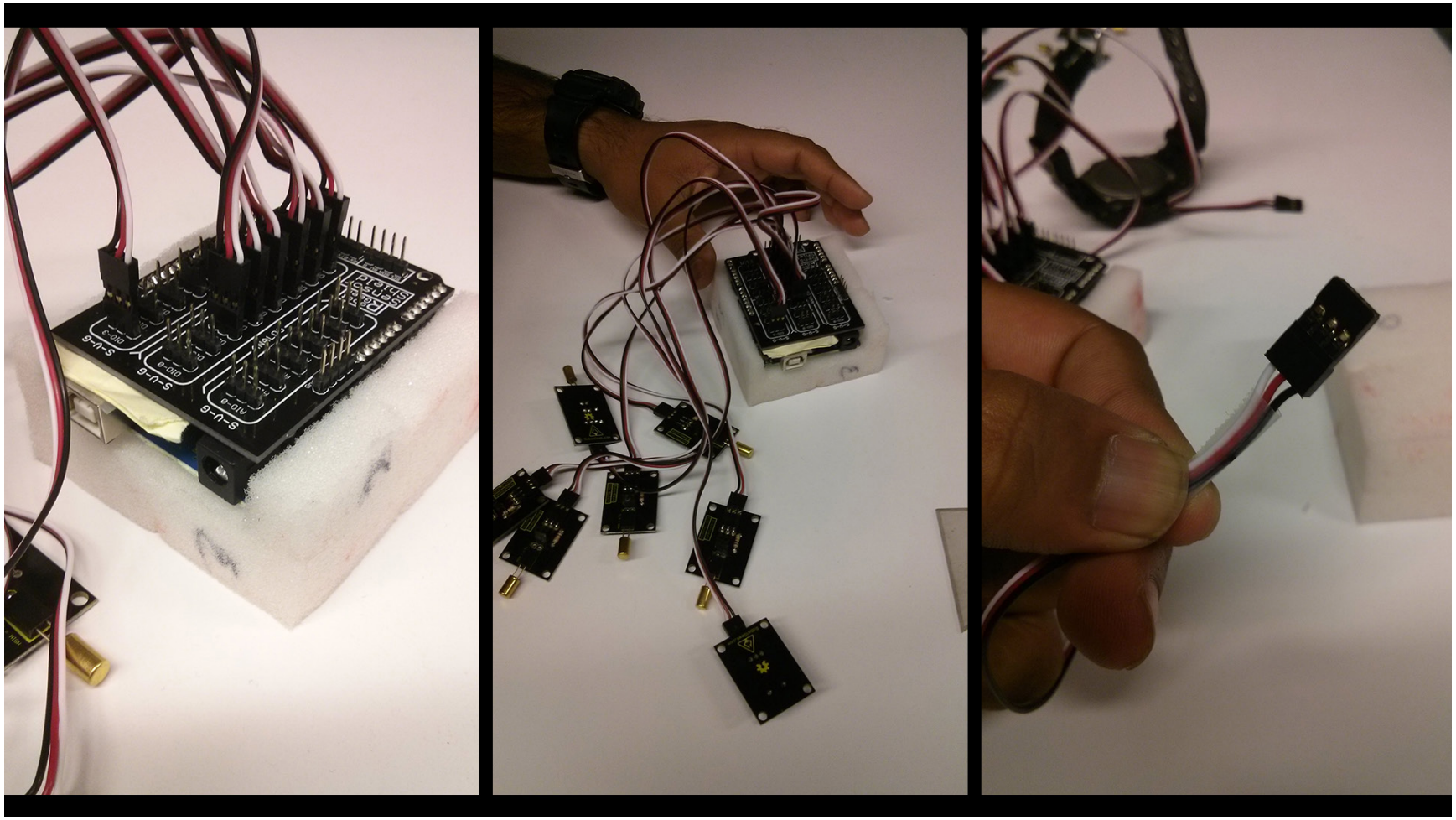
From the left to the right in picture 17, we were shaping the foam body of the dice in a measured manner so that Arduino board as well as sensor shield can be fit inside well.



Picture 17. Shaping Foam Body of the Dice

PROTOTYPING

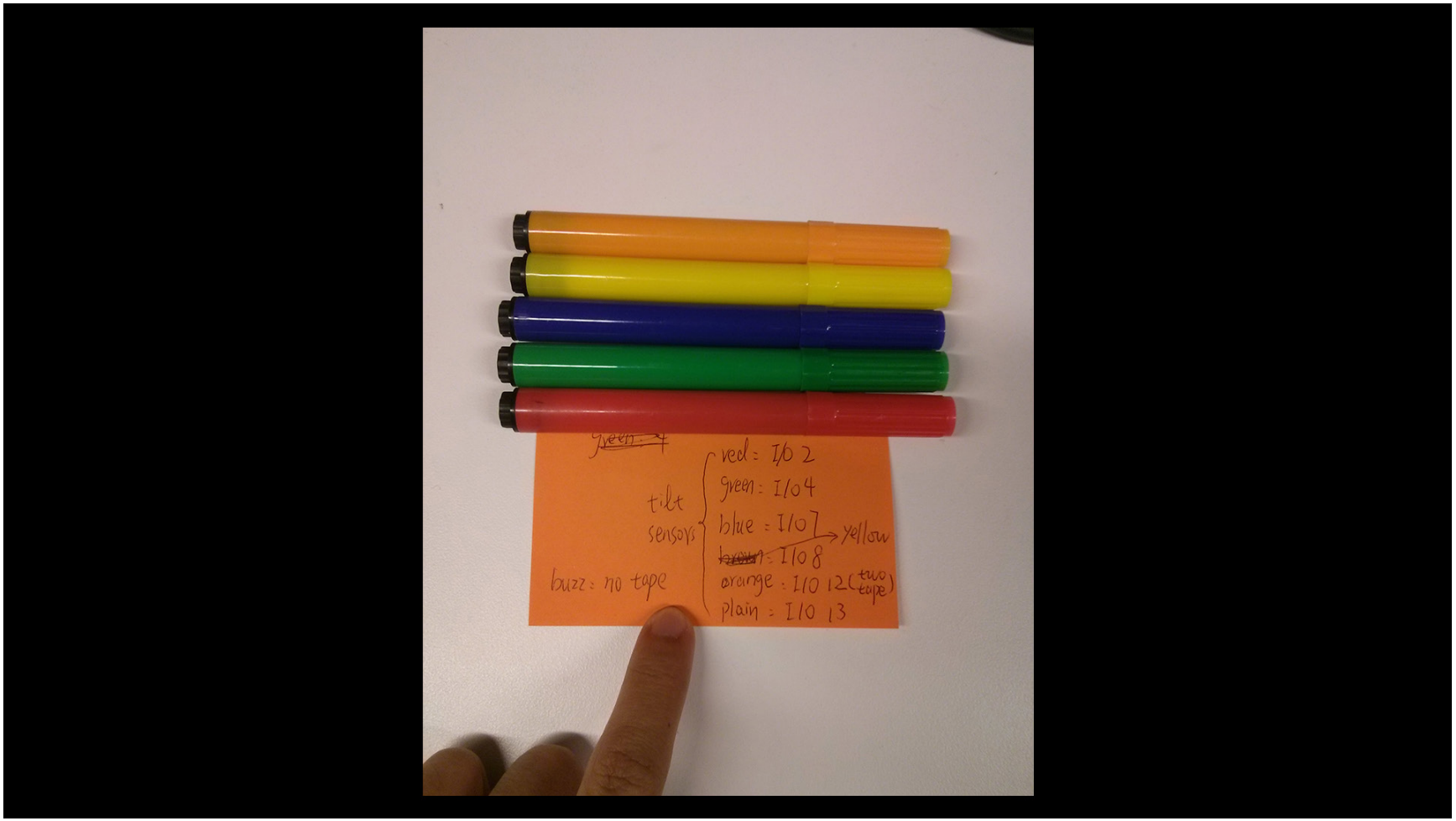
From the left to the right in picture 18, after we placed the board and the shield into the groove we cut, we were installing all the sensors and color coding each wire with color pens and scotch tape.



Picture 18. Installing Sensor and Color Coding Wires

PROTOTYPING

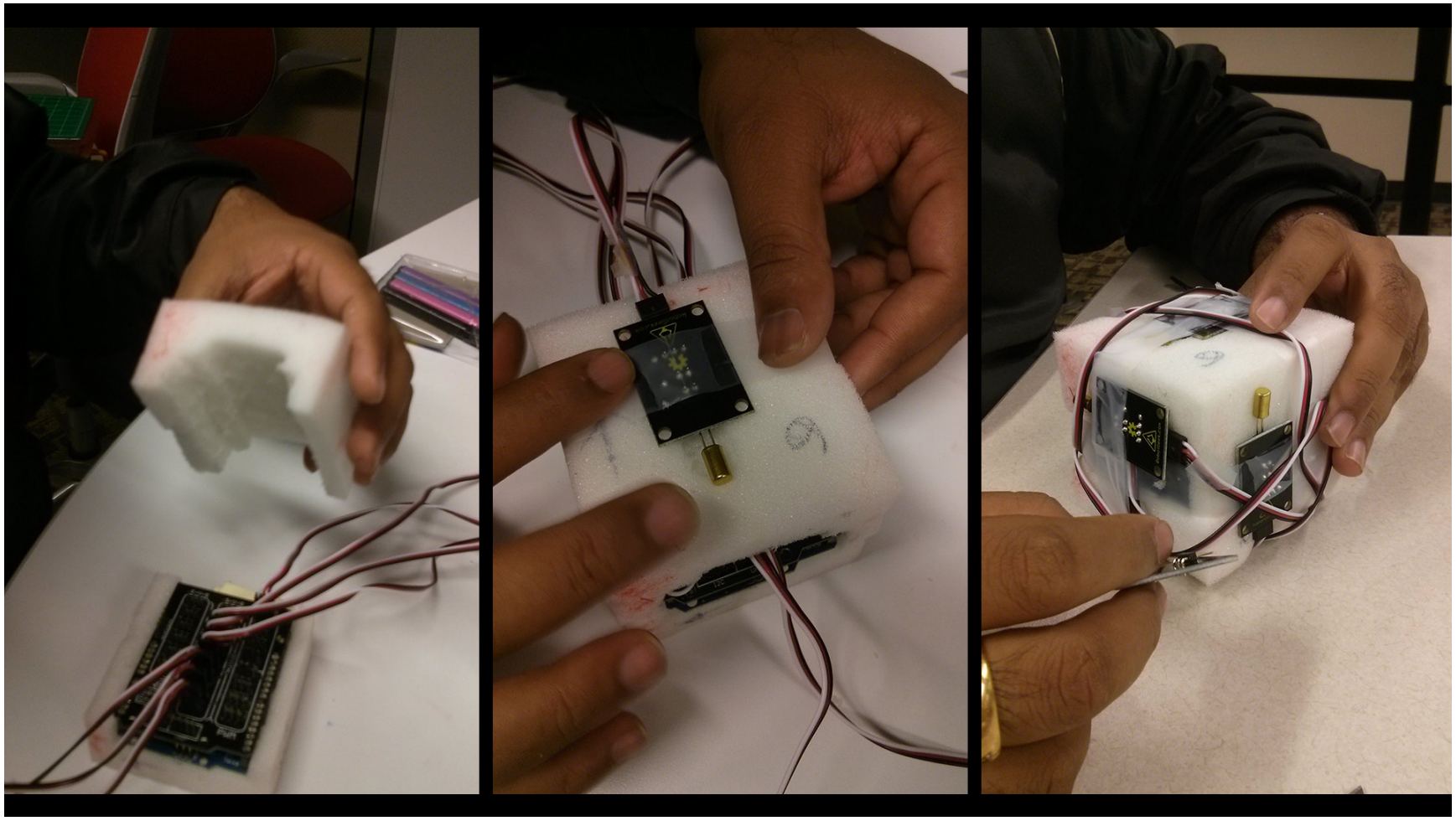
In picture 19, we used five different colors to mark five pieces of scotch tape separately. They were pasted on five wires respectively and the last one was left blank so that we were able to distinguish them.



Picture 19. Using Five colors to Mark Wires

PROTOTYPING

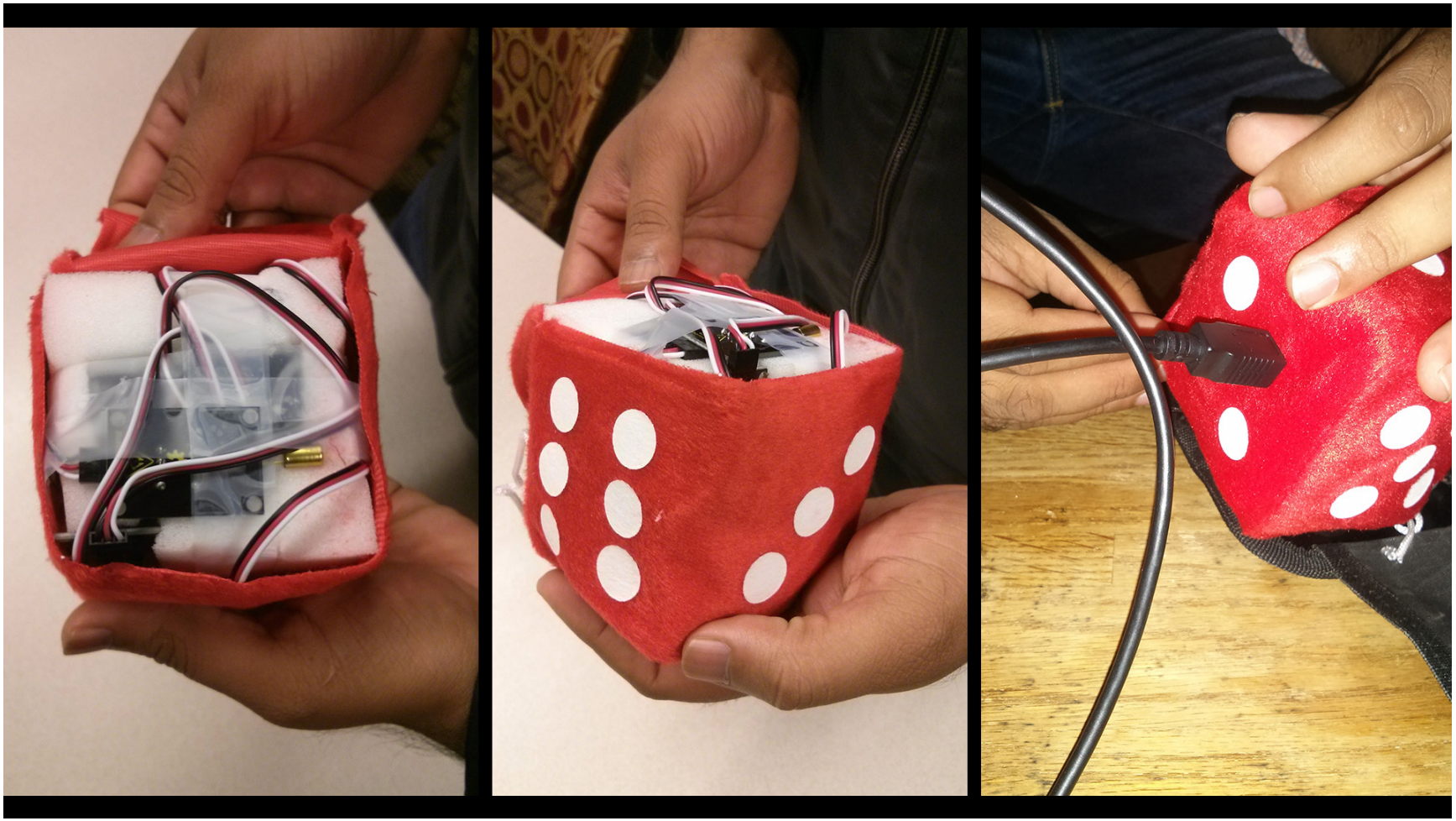
From the left to the right in picture 20, we were fixing all the tilt sensors and wires around the surface of the foam cube by applying scotch tape. It allowed us to alter the arrangement without irrevocable settlement.



Picture 18. Color Coding the wires

PROTOTYPING

In picture 21, finally this integrated interactive system was into the red dice cover. The opening was sealed by using safety pins. One small hole was cut on the side with figure 3 in order to plug in USB power port.



Picture 21. Final Encapsulation of the Dice

PROTOTYPING

Reflection on Issues

There were several issues showed up during the prototyping process. All of them were solved one by one in the end. We are going to talk about each problem and the ways through which we fixed them below.

Issue 1. Bad contact between Arduino motherboard and RobotGeek sensor shield.
In our early trials on testing the circuit which connects Arduino board, RobotGeek sensor shield and those tilt sensors, we found that the buzzer stopped working for some reasons. By trying to solve it in different ways, it turned out that use the larger female USB-B ports may cause a short circuit on the V and Ground pins of port DIO-3. To avoid this, we put a small piece of paper over the USB connector so that the metallic parts wouldn't be touching each other anymore.

Issue 2. The tricky placement of tilt sensors on the body of the dice while testing.
After we made sure that each electronic components work well, it was time to place them onto the body of the dice. During our attempt to make attachment, we thought about positioning them inside or fixing them on the exterior surfaces of the dice carefully. Based on how tilt sensor works, both way required us to put each sensors to be perpendicular to each side they were facing against with. This awareness made us decide to attach them externally so that it could be easier to make any changes afterwards. We finally used rubber bands to fix the tilt sensors temporarily and successfully. [image]

Issue 3. Distinguishing different sound patterns for representing different sides of the dice.
This was a basic concern for our prototype throughout the entire process. One piece of program which allowed the prototype to play six

PROTOTYPING

variant pieces of melody recurrently was produced for testing. It worked well but we still wanted it to play representative sounds like one to six beeps to map 1 to 6 numbers respectively. However, after the modification code had been made, we tested its performance and realized that no matter which side is facing up, playing beeps for only once may be hard for blind people to confirm what number it is after all, especially there might be a chance that beeping sounds become overlapped because of possible delay, So we asked ourselves questions like “what sound patterns are meaningful for representing numbers when you cannot see anything?” Then we looked back upon the previous program and came up with the final solution, which is playing the six representative sound patterns in a continuous loop like what we did for the testing.

Issue 4. The tricky placement of tilt sensors on the body of the dice while assembling.
The internal dice body is a foam cube in the right size that can accommodate Arduino board, sensor shield, tilt sensors and all the wires we were using. Just because we needed to put so many things all in one space, a plan for positioning each components properly was generated along the way. For placing the board and shield, we hollowed out the central part of the dice, leaving two opposite sides open. For fixing the six sensors, firstly we cut holes on each side of the dice according to sensor’s size and shape so that they can be fit into a relatively closed room, and then we attached them onto the body of the dice by using scotch tape. For organizing those wires, we adopted color-coded tapes on each wire in order to distinguish them while laying them over the entire surface of the dice body.

LESSONS LEARNED

What would you do differently if you were to do it again?
Our major challenge was the sensitive nature of tilt sensors, we would pick proportional tilt sensors which are way accurate the ball in a cage structure tilt sensors. Also, we would buy a bigger fuzzy dice, a 5 inch one, to organize and manage the wire effectively in the dice.

What would you have liked to have included?
Our initial plans were also to include vibration feature, i.e dice vibrate based on the number facing up. This is to extend usage of interactive dice for the people with both blind and deaf. Further, our plans were also include led lights instead of dots on the dice for the numbers, so that one can use the dice to play in the dark.

What would you like to be able to do with Arduino next?

We would like to play with wireless sensor transmissions, and use arduino as a medium to enhance the interactivity. We want to carry forward our project, and start working on series of product for the marginalized group of people we selected.

REFERENCES

[1] http://www.engineersgarage.com/articles/what-is-tilt-sensor?page=2

[2] http://www.trossenrobotics.com/robot-geek-sensor-shield?relatedid=1444

[3] http://www.trossenrobotics.com/robot-geek-tilt-sensor

[4] http://www.trossenrobotics.com/robot-geek-buzzer

APPENDIX

Basic Code Using Robotgeek Shield

```
#define TILT 2
#define BUZZER 3

//variables to hold the current status of the button.
(LOW == unpressed, HIGH = pressed)
int tiltState = 0;

void setup() {
  // set the pin for the Buzzer as output:
  pinMode(TILT, INPUT);

  // initialize the pins for the pushbutton as inputs:
  pinMode(BUZZER, OUTPUT);
}

void loop(){
  tiltState = digitalRead(TILT);  // read input value
  if (tiltState == HIGH)
  {
        // turn Buzzer on:
        digitalWrite(BUZZER, LOW);  //turn the buzz-
er on
  }
  else
  {
        digitalWrite(BUZZER, HIGH);//turn the buzz-
er off
  }
}
```

APPENDIX

Code with Three Tilt Sensors

```
#define TILT1 2
#define TILT2 4
#define TILT3 7

#define BUZZER 3



//variables to hold the current status of the button.
(LOW == unpressed, HIGH = pressed)
int tiltState1 = 0;
int tiltState2 = 0;
int tiltState3 = 0;


void setup() {
  // set the pin for the Buzzer as output:
  pinMode(TILT1, INPUT);
  pinMode(TILT2, INPUT);
  pinMode(TILT3, INPUT);
  // initialize the pins for the pushbutton as inputs:
  pinMode(BUZZER, OUTPUT);
}

void loop(){
  tiltState1 = digitalRead(TILT1);  // read input value
  tiltState2 = digitalRead(TILT2);  // read input value
  tiltState3 = digitalRead(TILT3);  // read input value

  if (tiltState1 == HIGH && )
  {
        // turn Buzzer on:
        digitalWrite(BUZZER, LOW);  //turn the buzz-
er on
  }
  else
  {
        digitalWrite(BUZZER, HIGH);//turn the buzz-
er off
  }
```

APPENDIX

```
        //check if the second pushbutton is pressed, and play a series of notes. Otherwise, nothing happens.
  if (tiltState2 == LOW)
  {
        //play first note
        tone(BUZZER, 3500,165);//the tone() function will generate a tone on pin BUZZER at frequency 3500, for 165ms
        delay(200);//tone() will set the buzzer for 165ms, but we still need to wait before we issue the next tone() command. If we wait a littl
        //play second note
        tone(BUZZER, 4000,165);
        delay(200);
        //play third note
        tone(BUZZER, 4500,165);
        delay(200);
  }
```

```
        //check if the second pushbutton is pressed, and play a series of notes. Otherwise, nothing happens.
  if (tiltState3 == LOW)
  {
        //play first note
        tone(BUZZER, 3500,165);//the tone() function will generate a tone on pin BUZZER at frequency 3500, for 165ms
        delay(500);//tone() will set the buzzer for 165ms, but we still need to wait before we issue the next tone() command. If we wait a littl
        //play second note
        tone(BUZZER, 4000,165);
        delay(500);
        //play third note
        tone(BUZZER, 4500,165);
        delay(500);
    }
 }
```

APPENDIX

Code with Six Tilt Sensors

```
#define TILT1 2
#define TILT2 4
#define TILT3 7
#define TILT4 8
#define TILT5 12
#define TILT6 13

#define BUZZER 3


//variables to hold the current status of the button.
(LOW == unpressed, HIGH = pressed)
int tiltState1 = 0;
int tiltState2 = 0;
int tiltState3 = 0;
int tiltState4 = 0;
int tiltState5 = 0;
int tiltState6 = 0;

void setup() {
  // set the pin for the Buzzer as output:
  pinMode(TILT1, INPUT);
  pinMode(TILT2, INPUT);
  pinMode(TILT3, INPUT);
  pinMode(TILT4, INPUT);
  pinMode(TILT5, INPUT);
  pinMode(TILT6, INPUT);


  // initialize the pins for the pushbutton as inputs:
  pinMode(BUZZER, OUTPUT);
}

void loop()
{
  tiltState1 = digitalRead(TILT1);  // read input value
  tiltState2 = digitalRead(TILT2);  // read input value
  tiltState3 = digitalRead(TILT3);  // read input value
  tiltState4 = digitalRead(TILT4);  // read input value
  tiltState5 = digitalRead(TILT5);  // read input value
```

APPENDIX

```
  tiltState6 = digitalRead(TILT6);  // read input value

  if (tiltState1 == HIGH)
  {
        // turn Buzzer on:
        digitalWrite(BUZZER, LOW);  //turn the buzz-
er on
  }
  else
  {
        digitalWrite(BUZZER, HIGH);//turn the buzz-
er off
  }



        //check if the second pushbutton is pressed,
and play a series of notes. Otherwise, nothing
happens.
  if (tiltState2 == LOW)
  {
        //play first note
        tone(BUZZER, 3500,165);//the tone() func-
tion will generate a tone on pin BUZZER at fre-
quency 3500, for 165ms
        delay(200);//tone() will set the buzzer for
165ms, but we still need to wait before we issue
the next tone() command. If we wait a littl
        //play second note
        tone(BUZZER, 4000,165);
        delay(200);
        //play third note
        tone(BUZZER, 4500,165);
        delay(200);


  }



        //check if the second pushbutton is pressed,
and play a series of notes. Otherwise, nothing
```

```
happens.
  if (tiltState3 == LOW)
  {
        //play first note
        tone(BUZZER, 3500,165);//the tone() function will generate a tone on pin BUZZER at frequency 3500, for 165ms
        delay(500);//tone() will set the buzzer for 165ms, but we still need to wait before we issue the next tone() command. If we wait a littl
        //play second note
        tone(BUZZER, 4000,165);
        delay(500);
        //play third note
        tone(BUZZER, 4500,165);
        delay(500);

  }if (tiltState4 == LOW)
  {
        //play first note
        tone(BUZZER, 3500,165);//the tone() function will generate a tone on pin BUZZER at frequency 3500, for 165ms
        delay(1000);//tone() will set the buzzer for 165ms, but we still need to wait before we issue the next tone() command. If we wait a littl
        //play second note
        tone(BUZZER, 4000,165);
        delay(1000);
        //play third note
        tone(BUZZER, 4500,165);
        delay(1000);

  }

        //check if the second pushbutton is pressed, and play a series of notes. Otherwise, nothing happens.
  if (tiltState5 == LOW)
```

APPENDIX

```
 {
      //play first note
      tone(BUZZER, 3500,165);//the tone() function will generate a tone on pin BUZZER at frequency 3500, for 165ms
      delay(500);//tone() will set the buzzer for 165ms, but we still need to wait before we issue the next tone() command. If we wait a littl
      //play second note
      tone(BUZZER, 4000,165);
      delay(200);
      //play third note
      tone(BUZZER, 4500,165);
      delay(500);

 }if (tiltState6 == LOW)
 {
      //play first note
      tone(BUZZER, 3500,165);//the tone() function will generate a tone on pin BUZZER at frequency 3500, for 165ms
      delay(200);//tone() will set the buzzer for 165ms, but we still need to wait before we issue the next tone() command. If we wait a littl
      //play second note
      tone(BUZZER, 4000,165);
      delay(500);
      //play third note
      tone(BUZZER, 4500,165);
      delay(200);

 }

}
```