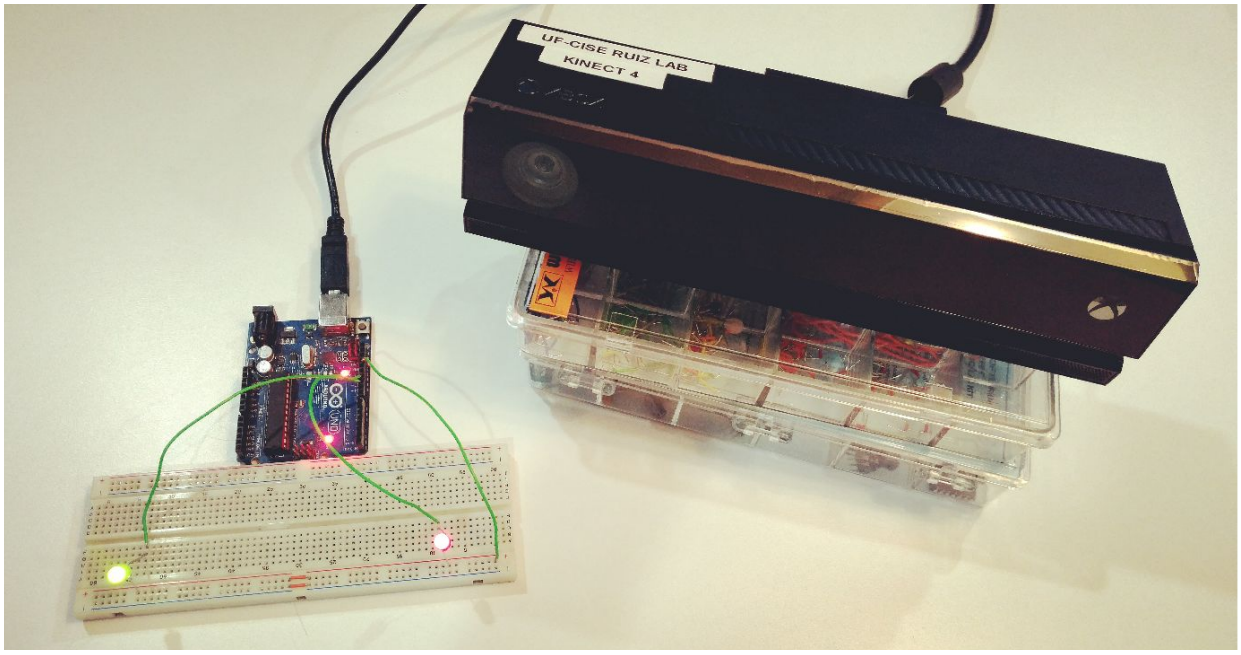
CEN 5726 Natural User Interaction

Course Project Final Paper

Group Members:
Siddharth Gupta
Manika Monga
Poornima Kumar
Dilip Kunderu

Section: Graduate

KINECTed LIVING



1 Iterative Design and Development Process

1.1 Initial Design

Our aim was to develop an interface that facilitated user's interaction with the system. We conducted a guessability study for the first prototype demo. Twenty participants of different age groups and experience with technology were asked to perform a gesture and give a speech command to accomplish the tasks we defined for our system as described below.

User Distribution	Age < 30	Age > 30	**Total**
Technical Users	9	3	12
Non-Technical Users	5	3	8
Total	14	6	**20**

Following is the list of tasks that the users performed.

- Switch ON the LED (left/right).
- Switch OFF the LED (left/right).
- Switching on both LED's
- Switching off both LED's

Based on the results of the study, we used the most common gestures and speech commands for our project. The first prototype included parallely implementing gesture detection and speech detection independently for the most common gesture and speech commands in the data set that was gathered from the users. The final step of the first prototype was to control(turn on/off) a single LED using a switch button on the arduino board.

Storyboard - Gesture

Storyboard - Speech

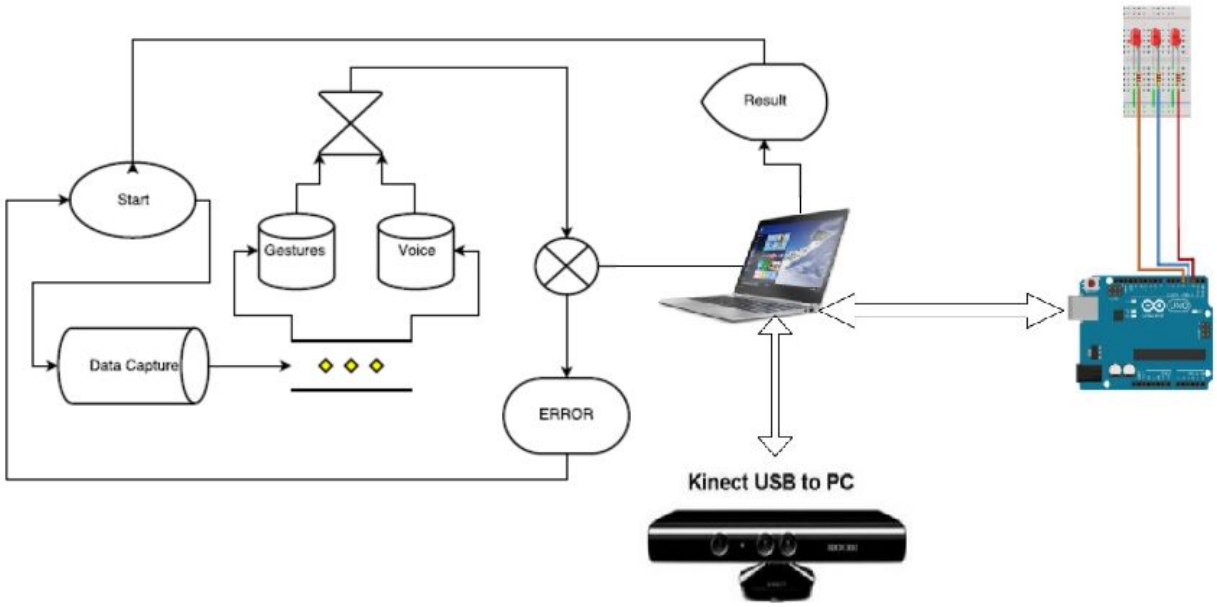
1.2 Developing the First Prototype

For the development of this project, we followed Agile Development. The development of the first prototype took place over the span of 3 weeks. In the first week, we familiarized ourselves with the software, downloaded the API's and Frameworks. We also collected all the hardware required to build the project such as the LEDs, the arduino board and wires. During the second week, the first user study was conducted to decide on the gesture and speech commands that were used for the development of the first prototype of the software. In the third week, we worked on building the software and integrating the application with the results of the user study. In the end, we performed Unit Testing, evaluating each component of the software.

We used various third party API's, toolkits and frameworks, while building the software. For gesture recognition, Microsoft Kinect Studio and Visual Gesture Builder was used. For speech recognition, we used Kinect Speech Platform SDK 11.0 and Microsoft Speech SDK Version 11. Visual Studio 2015 was also used for C# programming of the Kinect. Furthermore, we also used Arduino IDE v1.8.1 for programming the arduino board to control the LEDs. Version control and bug tracking was performed using regular code uploads on Github.

Each group member was assigned specific tasks for the project. Siddharth Gupta worked on the Arduino, Speech/ Gesture Recognition and Interfacing. Poornima Kumar worked on Gesture Recognition and building the User Interface. Manika Monga worked on the Arduino and Gesture Recognition. Dilip Kunderu implemented the Speech Recognition and User Interface.

The system architecture is as follows:



System Architecture (Separate implementation for speech and gesture)

1.3 User Study Summary

After the first prototype, the user study was conducted over a duration of 5 days ie., from 20th March to 24th March. Out of the 20 users, approximately 9 were female and 11 male. The user study was conducted at two locations in order to include people of all age groups and technical experience. Users were called in a group of 4-5 and were given a brief idea about the project and the tasks they had to perform. We also demonstrated the gestures and answered their queries. Each of the users were then asked to perform the listed tasks individually and in isolation to ensure that none of their activities and their feedback could be influenced by other users. Then, the users were asked few questions regarding the ease-of-use with our system and were also given a survey to fill.

From the user study findings we encountered a quite a few critical incidents:-

- Unless otherwise instructed where to stand, the user's body would sometimes be out of frame, or his arm would go out of frame while performing the gesture.

- The users sometimes also performed the gestures in a way that was not in accordance with the gestures in the database and thus sometimes the application was unable to identify the gesture which results in no output.
- Furthermore, the application sometimes struggled to detect commands if spoken in a slow speech rate, a different accent than expected by the system, or a low pitch.

Hence, the following are some of lessons we learnt.

- When the Kinect detects if a user skeleton goes out of frame, the user can be notified on such events with the help of a speech command asking him/her to come back in the frame.
- Furthermore, from the user study and the survey, we gathered that around 55% of the users preferred a synchronous implementation of gesture and speech commands, 25% users found Speech recognition to be more convenient, while only 20% preferred Gesture Recognition. Therefore, we decided that for the final prototype we will have a system where the users can give a combination of gesture and speech commands to perform the any of the tasks.
- Apart from expanding the dictionary for speech commands, multiple gestures with slight nuances for a single command can be added to the database.

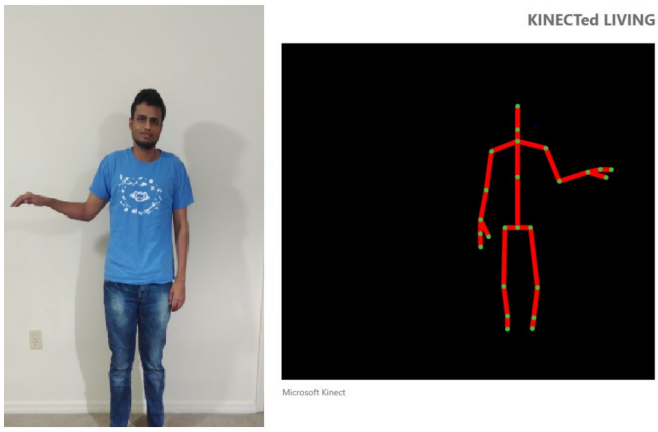
1.4 Developing the Second Prototype

From the critical incidents found while conducting the user study on our first prototype, there were many lessons to be learnt and implemented to the project. These included:-

1. Resolving ambiguous gestures performed by the users
2. Brightness Control
3. System Integration (combining speech and gesture application)
4. Feedback provided to the user

1.4.1 Change 1

The users sometimes performed the gestures in a way that was not in accordance with the gestures in the database. The problem was severe as the application in such instances was unable to identify the gesture which results in no action. 3 users bent their arms while performing the gesture. The gesture as defined in the gesture database required full extension of the arm to be identified correctly by the application. As seen in the image below the user had the arm bent while performing the gesture whereas our system expected a full extension of the arm.



Hence for the final prototype, we thought about improving the gesture detection by creating slightly nuanced versions of gestures for a single command and added to the database. This allowed the application to recognize ambiguous gestures and functions as desired.

1.4.2 Change 2

Some users found the functionality implemented in the initial prototype limited to only turning on/off the LED's and suggested improvements. Seven users had suggested the implementation of brightness control to our system. Therefore, we decided to implement brightness control in our final prototype. We had thought of implementing three levels of brightness: High, Medium and Low, which could be controlled by both gesture and speech commands. To include brightness control, we had to come up with new gestures and speech commands, since this feature wasn't developed in any of the earlier prototypes. We started running into issues while deciding on the

new gestures for brightness control since any action with the left or right arm was conflicting with the gestures already in place to switch on the LED's. To overcome this, we had to train the gesture database really well for the previous and the new gestures and then filter out any overlappings by using confidence scores of the gestures as a measure. Similar to this, we faced issues for the speech commands as well. The keywords such as left, right would get detected for turning on as well as changing the brightness of the LED's. We had to rewrite the speech grammar to resolve this and additionally filter out overlappings by using confidence scores again.

1.4.3 Change 3

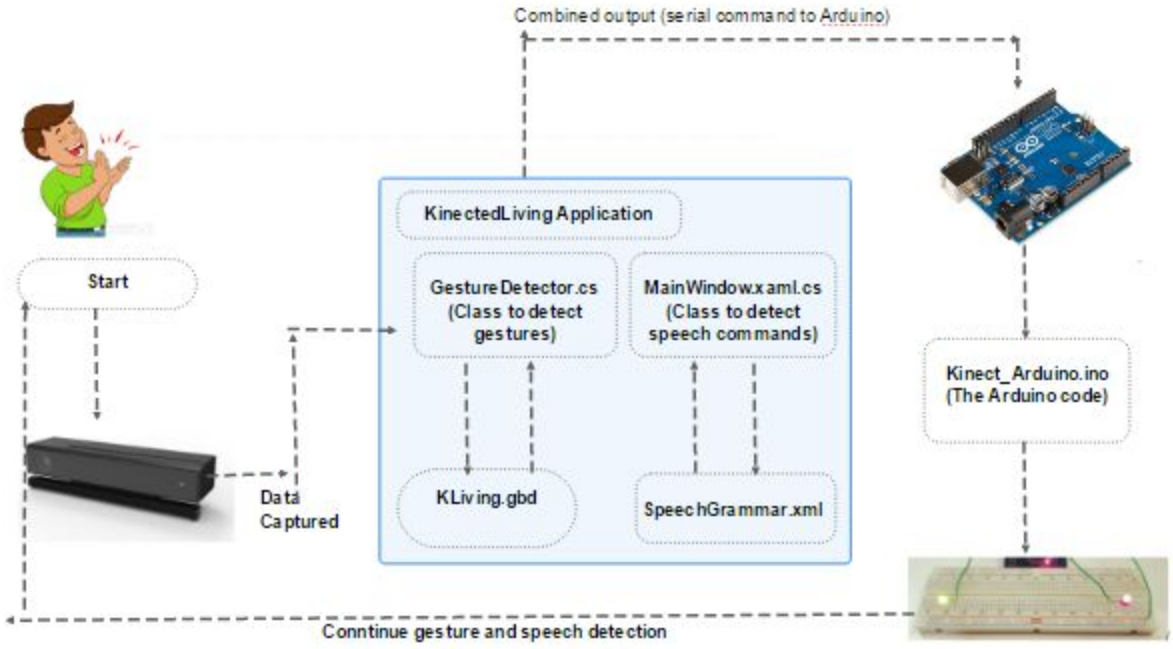
From the user study and the survey, we gathered that around 55% of the users preferred a synchronous implementation of gesture and speech commands, 25% users found Speech recognition to be more convenient, while only 20% preferred Gesture Recognition. We observed this result was due to the fact that the gesture + speech combination was the most natural way of interaction with our system. On the other hand, our users told us that they wouldn't really want to perform gestures since they felt shy or to them the speech commands seemed much more convenient. Therefore, for the final prototype we decided to have a system where the users can give a combination of gesture and speech commands to perform the any of the tasks. The gesture would aid in selection of the device and the speech would be responsible for changing the state of that device. Apart from this, the users could continue to perform tasks using either of the speech or gesture commands solely. To achieve this we needed to integrate the separate applications we had created earlier and merge them into one. However this was easier said than done. We had to carefully identify how the methods in each program were called and declare and initialize sensors and other resources appropriately.

1.4.4 Change 4

We initially had a computer speaker speak out if the action was performed by the system when the user gave a speech command. For instance, if the user asked the system to "Turn on the left LED", the system would respond with "I have switched on the left LED". Similar was the case for the other commands. However, when we added the functionality for brightening and dimming the LED's the system would go into self infinite loops since it would detect certain keywords from what the computer spoke out and assume that it was the user's voice. As a result, it would repeatedly keep on performing the same command again and again, and subsequently speaking out the feedback phrases. We had a hard time resolving this. We tried several permutations of feedback phrases, however it still didn't work well with 100% certainty. Finally, we decided to replace all feedback phrases with just a simple word 'DONE'.

2 Final Architecture

2.1 System Architecture



2.2 Code Modules

The following classes implements the functionality for both gesture and speech detection:

1. GestureDetector.cs

   This class is responsible for the gesture detection and sends corresponding input commands to the Arduino Board.

   Methods:

   - public class GestureDetector : IDisposable :- The constructor initializes the gestureDatabase variable which stores the path to the gesture database that was trained with VGB. The following table gives the different discrete gestures in the database that needs to be tracked.

Variable name	Stored Value
leftledGestureName	ledON_Left
rightledGestureName	ledON_Right
allledoffGestureName	All_ledOFF
leftledlowbright	Lowbright_Left
rightledlowbright	Lowbright_Right
leftledhighbright	Highbright_Left
rightledhighbright	Highbright_Right
leftledpoint	point_Left
rightledpoint	point_Right

   - public GestureDetector (KinectSensor kinectSensor, GestureResultView gestureResultView):- This method adds all the gestures from the gestureDatabase.
   - private void Reader_GestureFrameArrived(object sender, VisualGestureBuilderFrameArrivedEventArgs e) :-

     This method helps in identifying the gesture of the tracked body that is detectced from the Kinect sensor. The gesture is compared to the gestures defined in the

constructor. When a tracked gesture matches a gesture defined in the gestureDatabase and if its confidence value is above a certain threshold then commands (turn on/off LEDs) to the Arduino Board is sent for the detected gesture.

2. KLiving.gbd :- The name of the gesture database file that was trained with VGB

3. MainWindow.xaml.cs

   This class is responsible for the speech detection and sends corresponding input commands to the Arduino Board. The class makes use of the SpeechGrammar.xml file for the detection of the speech commands.

   - public MainWindow():- The constructor initializes the port name as COM5 where the Arduino gets connected to and initializes BaudRate to 9600
   - private void SpeechRecognized(object sender, SpeechRecognizedEventArgs e):- This method detects the speech commands and if the confidence is above 0.3 then it sends the corresponding serial input to the Arduino. Once the Arduino command is executed, it gives a feedback in the form of speech which is implemented using SpeakAsync() task.

4. SpeechGrammar.xml

   The SpeechGrammar.xml file contains the different grammar rules for the speech commands. Multiple items (commands) are defined for each of the user task. This gives the user the flexibility to give different commands and perform the action.

Kinect_arduino.ino (Arduino code):

Kinect_arduino.ino file defines the pins for the LEDs. In this project, we have used pin 10 and pin 11 of the Arduino for left and right LED respectively. To switch on the LEDs, 256 is sent as the serial input for left LED and 257 is sent for the right. To turn off both the LEDs, 500 is sent as the serial input. There are two levels of brightness defined: 33%

(low brightness) and 100% (high brightness). Thus, this file is responsible for switching on/off the left and the right LED and increasing/decreasing the brightness of the LED. The code has to be uploaded to the Arduino before the port connection is made from the MainWindow.xaml.cs file.

2.3 Third-Party Tools

Visual Studio 2015	For C# programming of Kinect	https://www.visualstudio.com/vs/older-downloads/
Kinect Studio v2.0	Building the gesture database	https://developer.microsoft.com/en-us/windows/kinect/tools
Arduino IDE v1.8.1	Programming the Arduino Board to control the LED's	https://www.arduino.cc/en/Main/Software
Kinect SDK Browser	Discrete Gesture Basics & Speech Basics	https://www.microsoft.com/en-us/download/details.aspx?id=44561

2.4 Source Code

The following is the link to the project repository on GitHub:-
https://github.com/pkumar07/KinectedLiving

3 Future Work

Kinected Living as it stands today is a fully functioning prototype that showcases our home automation concept using a Kinect. We have improved our application incrementally acting upon the user feedback at each stage. We have been able to integrate the discrete speech and gesture applications into one and also include brightness control of the LED's. However, below is a list of features we would include in our prototype to make it a viable application.

1) *Detecting if a user skeleton goes out of frame* - The user can be notified on such events with the help of a speech command asking him/her to come back in the frame. This type

of feedback will help the user identify the whether his/her body fits in the frame while performing the gestures.

2) *Free speech interaction* - We have currently expanded the speech grammar of Kinect according to users suggestions. However we wanted to allow the users of our application to not be bounded by certain keywords/phrases. Hence we looked into building a solution by integrating speech recognition and feedback with Microsoft's LUIS (Language Understanding Intelligent System). However, since Kinect's method of detecting speech was to detect only those words that were made a part of its grammar, we realized integrating our system with LUIS wouldn't be fruitful. Hence alternative methods to allow free speech interaction could be explored for the finished product.

3) *Real world application* - To show the proof of concept, we demoed our prototype using two LED's. However, in a real use case, our application would control home devices. We can make this transition easily by using some additional hardware equipment like a 5 Volt relay for each device that needs to be controlled in a living space.

4) *Heterogeneity* - as a requirement was pressed upon by many of the users. A further extension to Kinected Living would be to include control of additional devices such as fans, Air Conditioning, etc.