LED Coffee Table

Edward Lai, Jin Kim, Matt Miller

Dept. of Computer Science, UC Berkeley. Email:
{edward.lai, jinhunkim, mbmiller}@berkeley.edu

1. Introduction

This goal of this project was to develop a usable household device that would incorporate an embedded system while maintaining the original functionality. The final design that was agreed on was to develop a coffee table that would respond to multiple kinds of sensor inputs and create varying visual displays as a result of those signals. The table utilized a glass top that allowed the passing of infrared waves to pass through and detect the motion and presence of objects on top of it. Beneath the glass was an LED array that would light up in various ways in response to the signals transferred to the mbed. The code is decoupled from the signal type so any type of sensor can be used as long as it sends a digital signal into the mbed board.

2. Hardware Design

The forefront of the design phase was to somehow create a method of displaying the LED NeoPixels while at the same time creating a surface that objects could be placed on top of. After researching a variety of sensors and similar projects, we came down to two options on how to solve this issue. The first approach was to use a solid top and to place cameras either at the perimeter of the table or in the room that the table would be placed. It would then be possible to determine whether or not an object was on the table by processing the footage. Additionally, it would be able detect motion on top of the table and indicate to the code how to respond. However, this was determined to be too computationally intensive for the mbed board. This would require an external server to handle the calculations and then to determine which NeoPixel lights should be turned on or off.

The other approach was to use IR sensors with a glass table top. When used with large wavelengths, in this case 940 nm, infrared signals can pass through translucent objects. This meant that we were able to send the signals through the glass table top and that they would then reflect off any objects that they hit. The downside to this approach was that dark objects had the potential of absorbing the IR waves and as a result the IR receivers would not notice the existence of an object. Ultimately our team decided on using the IR sensor approach because it did not require an external server and it was possible to embed the IR emitters and receivers within the LED NeoPixel array. This would later allow the software to have an easy time determining which LEDs would need to be turned on based on location relative to the IR receivers that sent the signals.

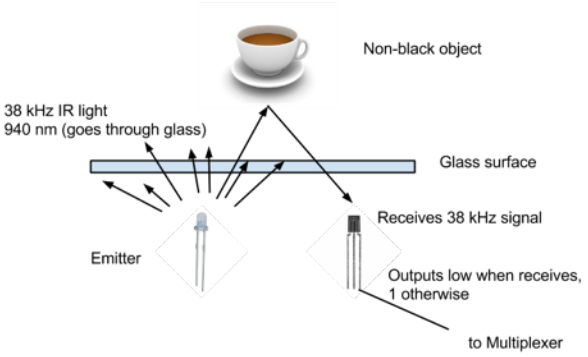

Figure 1 - IR sensor approach to detecting

The next design choice was how to wire all the different components. The IR emitters were easy to hook up: just put them at set intervals in the LED array and supply them with the same PWM signal. The NeoPixels were handled in a similar fashion with the only difference being that a control pin had to be passed from the mbed board to issue signals for the various light patterns. The difficulty came with how to group the different IR receivers to find the most efficient way of processing the data. Because there weren't enough pins on the mbed board to connect all the IR receivers directly we were faced with a choice of acquiring multiple mbed boards to accommodate the extra receivers or using multiplexers in order to consolidate the different signals into one pin. While another mbed board had various advantages such as providing more power and allowing for faster response times, it needlessly wasted pins. Furthermore, if the timing was off sync between the two boards, it was possible, although unlikely, for undefined behaviors to occur. On the other hand, multiplexers enabled us to use only one mbed, thus eliminating timing and saving

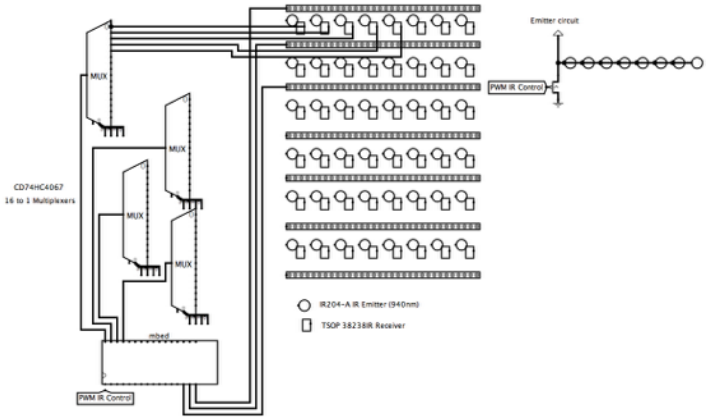

Figure 2 - Wiring of LED array and IR sensors to mbed board using multiplexers

pins. However, this approach would require us to cycle through all the different channels on the multiplexer in order to find whether or not an IR receiver detected an object or motion. This would lead to slight time delays in terms of responsiveness. Additionally we were uncertain if we could generate enough power for the entire LED array utilizing only one mbed board as the power source. In the end, we decided to go with the one mbed and multiple multiplexer approach in order to ensure that all components were operating on the same clock.

3. Software Design

For our project we needed the software to be able to handle incoming signals that represented the IR receivers detecting objects or motions while at the same time being able to produce commands for the NeoPixels in order to respond to the inputs. The main decision point here was whether or not we wanted our code to run serially or in parallel. Under the serial approach all our code would run on a single thread guaranteeing that it would have sufficient resources and would avoid any concurrency problems that may or may not occur. This would be the simplest implementation and would allow the mbed board to respond very quickly to a detected signal by activating the light within the block of code that detected the IR receiver signal. However, if this were to occur for multiple IR receivers at the same time the code would generate some delay when multiple objects were placed on the table at the same time and all the corresponding lights turning on as well. There is also the problem that this implementation would have a hard time processing more complicated light patterns which require state. This would require additional calculations between each read of the sensors making the overall code slower and less responsive.

The second approach that we considered was a two thread implementation. The first thread would be in charge of handling the incoming IR signals and update an internal representation of the board where objects were located. The second thread would then read the contents the internal representation and determine which lights to turn on and modify. This approach would allow a faster read time of the IR receivers allowing the code to approach real time reading of the signals faster. Furthermore it would be possible to group together changes and to send out batch changes as opposed to individual ones that we would find in the serial version of the code. However, this implementation creates a problem where there can be some delay between the first thread updating the internal representation and the second thread recognizing that an update had occurred and generating the appropriate NeoPixel commands. This delay would be increased with more complicated patterns resulting in the same problem as in the serial version of the code.

The last implementation that we considered was to have one master thread in charge of reading from the IR receivers and to spawn a new thread whenever an object or motion was detected. Each thread would be in charge of all calculations in regards to light patterns in regards to the object or motion that spawned it. The thread would then either kill itself when the light pattern was finished or the master thread would kill it when the object that spawned the child thread was removed from the board. This implementation minimizes the gap between IR receiver signals and light patterns being displayed and supports complicated patterns as well. However, this approach requires memory for each thread in addition to additional resources being used to store a table linking each signal to it's corresponding thread. Furthermore there could be conflicts between two threads over how to change an LED light and this could cause undefined behavior but could be remedied with an arbitrary method breaker. Ultimately the second approach with two threads was chosen due to better response times than the serial version of the code and uncertainty over how many threads the mbed could support in addition to memory constraints.

Under this implementation we have one thread loop through the four multiplexer signals and loop through the sixteen channels of each multiplexer. Upon detecting a signal it will immediately update the internal representation of the board and continue to loop through the IR receiver signals. The second thread then reads from the internal representation of the IR receiver signals and determines which NeoPixel lights need to go on. It can then update the internal representation that keeps track of which lights are on and off. It then proceeds to generate the proper signals and sends these out. Afterwards, this thread can look for any multistep patterns in the internal light representation and update the corresponding signals appropriately as well.

4. Results

In our final implementation we were able to create a table of 224 lights using 56 IR emitters and receivers. The table was able to display a myriad of colors and was able to accurately detect objects on top of the table through the glass top. The table was able to detect everyday mugs and notebooks with decent accuracy along it's surface. The only exception to this was in the presence of black objects that absorbed the IR waves instead of reflecting them back. However, we found in certain lightings and angles of the objects the LED display was able to pick up on the objects as well. The table was also able to detect motion roughly one foot above the glass surface of the table.

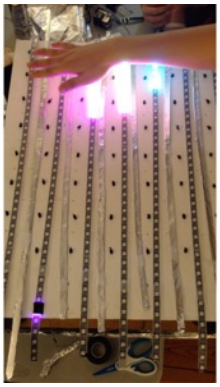

Figure 3 - LED NeoPixels responding to hand over IR sensors

This was an impressive accomplishment due to the accuracy of the table at the given distance. The table was able to detect only the suspended object and only lit up beneath it, meaning that the surrounding areas did not light up. This was a major accomplishment given that the IR emitter and receiver density was high enough that there was no part of the board that was unable to detect objects on top of it. Despite this high sensor density, the sensors did not overlap domains but instead remained distinct. Another noticeable property of the final implementation was the response rate of the mbed board. Although the chosen implementation was not considered the fastest in terms of response speed in comparison to some of the other implementations, there was no noticeable lag during the testing phase of this project. This in particular exceeded our expectations as we feared delays would interfere with the smoothness of motion based effects.

5. Problems

While creating this project we faced a variety of problems. The first of these problems was a result of the IR receivers that we chose to use. These receivers detected 38 kHz signals, which was standard for tv remotes and other devices that use infrared waves. Because of this the board would display erratic patterns whenever infrared signals were detected by the IR receivers aside from those generated by the IR emitters within the board. Although not done in this project, a simple solution to this problem would be to swap out the IR receivers with ones of another frequency.

Figure 4 - LED Coffee Table reacting to remote control IR signals

The second issue that we faced was the range of the IR emitters. The emitters sent out waves in all directions and as a result had a tendency to hit the IR receivers directly without being reflected off of any other objects. This would result in the program thinking that the entire board was constantly covered and result in the whole board always being lit up. This problem was remedied by placing folded aluminum foil strips between the IR receivers and the IR emitters high enough to block direct IR signals from hitting the IR receivers but short enough to not interfere with reflected waves.

Another issue that we encountered while working on this project was with the multiplexers. The multiplexers were not ideal and would occasionally blur two different signals resulting in the program believing that their were objects on the table that did not really exist. This problem only occurred in a few multiplexers though, so this was a hardware issue and not easily solvable. However, the multiplexers also created a problem when interacting with the code on the mbed. The mbed would send certain selector bits to the mux, hoping to read from the corresponding channel. However, the rate at which the multiplexer was able to return the corresponding data was significantly slower than the codes query rate so the multiplexer ended up returning data from other channels to the mbed. This problem was solved by adding a 0.001 second delay to the code which gave the multiplexer enough time to properly generate the signal containing the channel information.

The final issue we encountered had to do with the power supply. Although we had earlier opted to use one mbed instead of two in order to prevent issues with different grounds we ended up noticeably overdrawing our power supply. Although everything

was able to function, the NeoPixels became noticeably dimmer as we added more and more rows of LEDs, IR emitters, and IR receivers. This also extended to the IR emitters, where the strength and distance of that the IR waves travelled dropped as more components were added to the circuits. This issue could have been easily fixed with an additional power source but was not done within the bounds of this project.

6. Lessons Learned

While working on this project the team learned many key concepts about working with embedded system designs. The most important of these was the criticalness of the design phase in conjunction with prototyping. Although during the design phase we came up with multiple approaches to both the hardware and software sides of this project we only reasoned through them at a theoretical level before settling on which design we wanted to follow through on. Even the one prototype that we developed only served to prove that the IR signals were capable of passing through the glass. This resulted in us prematurely making some design decisions. For example we had fears of lag with the two threaded implementation of the software. However, had we prototyped this, we would have seen, as with the final product, that this would not be an issue. Knowing this, we could have explored similar implementations without fear of display latency and also tested the memory bounds further as well. Although creating a prototype for the video sensor as opposed to IR sensors may have been difficult and time consuming there may have been valuable insight in there as well.

Hardware considerations were another aspect of the project that our team did not properly assess. As a whole we vastly underestimated the time to assemble and debug the circuitry of the system. This problem was exasperated by us not acquiring spare parts to replace broken or glitchy gadgets within our product. This ended up with us having to reduce the size of the final product to accommodate the number of working components that we had. Also the increased time spent in assembly and adapting to unexpected hardware bugs greatly limited the time we could spent on software development. This resulted in simplifying light patterns as well as being unable to achieve all of the goals set at the start of this project.

The last thing we learned was power and logic flow throughout the project. On the hardware side it would have been beneficial to the team if we had calculated the power consumption of the NeoPixels and IR sensors and compared it to the mbed board power supply. As previously noted we experienced a drop in performance within our components as a result of not having enough power. Had this been previously calculated it would have been possible to switch to less power consuming alternatives or to obtain additional power sources. Within the software more formal analysis would have greatly aided in run time calculations. This would have helped in choosing the appropriate implementation as well as whether or not other approaches should have been considered. This is especially the case in regards to how many and the use of threads within the project. Taking into account average use case as well as the extremes would have been more concrete and would have served as a better metric for selecting a software design.

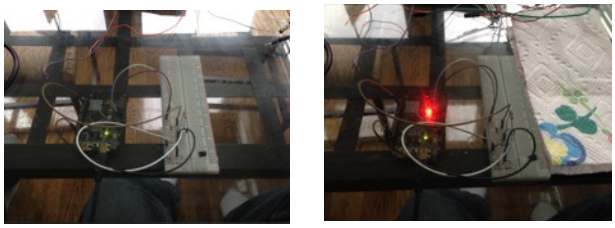

Figure 5 - Hardware Prototype displaying IR sensors working through glass table

7. Next Steps

This project serves as a baseline from which a lot of improvements can be made. The most basic improvement that can be made is to increase the NeoPixel and IR sensor density in order to improve accuracy of the sensors. This would enable more precise and complex patterns in response to a greater variety of actions. Other simple features that could be expanded to would be the use of different kinds of sensors. Because the code is decoupled from the kind of sensor being used it would be possible to use pressure, temperature, and audio sensors in order to activate the light patterns as well. This would open up many ways in which the LED coffee table could interact with the environment. Along these same lines, more complicated light patterns could be implemented as well. This could include ripple effects from objects. This would require some modifications to the code to maintain previous states as well as calculating new light locations based on previous light arrangements.

More fundamental changes to the design could also be made in order to improve the final project. An external power source would be able to overcome some of the problems listed. Also a more complex platform would allow the support more extensive software implementations. For example, the multithreaded implementation with a master and many child threads would be able to support much more complex patterns. Furthermore it would be possible to develop interesting games such as Tetris on the LED coffee table. This would be possible by tracing the hand movements with the IR sensors to determine how the player would want to manipulate the falling blocks. Overall the LED coffee table developed serves as a good base but leaves room for many different kinds of improvements and expansions.