Slide 1:

Good morning teachers, guests and fellow Dunmanians. I am Bryan Leow, and today I’ll be presenting on the research project that I conducted last year, entitled “Designing a Safekeeping System with Human Touch Detection Module”. It was conducted under the Research @ Young Defence Scientists Programme, or R@YDSP in short, and I’ll be covering more about this programme in a moments’ time.

Slide 2:

But first, let’s take a look at the presentation overview and what we can expect to cover today. I’ll first talk about the background information you’ll need to understand this project, followed by a brief covering of my methodology. We will then proceed to results and discussion, before wrapping up with some miscellaneous information about Research @ YDSP.

Slide 3:

We will first proceed head on into the background info.

Slide 4:

So after reading my project title, you would most likely be thinking: what does “human touch detection module” even mean? To put it succinctly, it refers to the act of using capacitive sensingof the human body to trigger an alarm; and this capacitive sensing is made possible due to a property of the human body that enables it to act as a capacitor.

Some you out there are probably wondering that a capacitor is, so here’s the definition. Basically, it’s an object that can store electric charges, and this black cylindrical thing you see on the screen here is your typical capacitor.

With this definition of a capacitor, can anyone of you here guess what property of the human body enables it to act like one and store electric charges?

This property is the ability of the human body to conduct electricity! Because electric charges can flow through our bodies, we can also store these charges in us just like this capacitor here, and this gives rise to the concept of capacitive sensing.

Slide 5:

So now that we know what capacitive sensing is, we must now go about building a sensor that utilises this mechanism. To do this, we could use the Arduino microcontroller, which functions somewhat like a computer only with much processing power. An Arduino microcontroller is able to perform capacitive sensing using a resistor-capacitor network, formed by a resistor and a touch electrode capacitor connected to the Arduino, which is the aluminium foil.

Slide 6:

Pictured is the Arduino microcontroller. It consists of many pins that can either send or receive information. In this case, pin 2 is configured to be the send pin, and is connected to pin 6 and the resistor. Pin 6 is the receive pin, with is connected to pin 2 and the aluminium foil.

Slide 7:

In the first step, a 5 volt current will be passed through the send pin, which is termed toggling the send pin to the HIGH state. The Arduino microcontroller will then wait for this current to flow to the receive pin and make it toggle to the HIGH state as well. How long the microcontroller has to wait for this process to occur is given by this RC time constant equation.

R refers to the resistor value, Cpin is the capacitance of the aluminium foil, and Ctouch is the capacitance introduced by the interfacing object on the aluminium foil, which is the hand of an approaching person in this case.

Slide 8:

Once both pins are at the same state, the current through the send pin will be cut and the pin will be toggled to the LOW state. We will then wait for the receive pin to do the same and discharge, before repeating steps 1 and 2 again.

This causes the receive pin to be periodically charged and discharged, and the frequency of this process is dependent on the RC time constant. With a larger RC time constant, the pin will take longer to charge and discharge and will hence do so less frequently.

Pictured is a graphical representation of this occurring. The black line represents the state of the send pin and the pink line represents that of the receive pin. When the send pin toggles to the HIGH state, you can see that the black line moves vertically upwards while the pink line increases more gently. This is, as previously established, caused by the time it takes for the receive pin to charge and get to the HIGH state. Eventually both will reach the HIGH state as seen by this horizontal part. When the send pin toggles to the LOW state, the receive pin will similarly discharge gradually back to the LOW state. Hence, these ups and downs on the graph represent the continuous charging and discharging of the receive pin.

Slide 9:

So how does this mechanism help detect human touch? Basically, a touch from an electrical conductor such as a hand will increase Ctouch. This Ctouch increase is extremely small, but since Cwill be multiplied by R based on the RC time constant equation, this small increase will be amplified by one million times. Hence RC time constant increases noticeably, and this increase will be reported to the Arduino.

Going back to the graphical representation, we can clearly see the difference when there is no touch, pictured above, versus when there is human touch causing overall capacitance to increase, pictured below. The time it takes for the receive pin to toggle to the HIGH state is much larger with human touch than without it.

From there, we can write a program to make the Arduino sound an alarm, if this increase satisfies a certain condition as defined in the program we wrote. For instance, we could set a threshold value of say 30, and if the RC time constant exceeds 30, it would sound the alarm. Or, we could measure the increase in the RC time constant over a given period of time, and if it increases by say 15 within 5 seconds, the alarm would sound.

But which method is better for the system? And what RC value should we be using as the trigger value?

Slide 10:

This brings me to the second segment of this presentation, the methodology.

Slide 11:

As already established, we will need to evaluate if the alarm should be triggered when the RC time constant exceeds a predefined threshold value OR increases by more than a certain value within a period of time. This will be termed experiment 1, and the first method will be called method 1.

After this, we will then need to find the best value to use in our sensor, in experiment 2.

Slide 12:

For experiment 1, separate programmes were written for both possible designs. The code you see on the screen is a small segment of the programme I wrote for method 2, which is to measure the increase in the RC time constant within a period of time.

The important part of this code is the “if” function under void loop. If all three conditions as listed here are satisfied, the Arduino will execute this chunk of text below which plays the alarm, else it will skip it entirely. The condition I will focus among these three is the centre one, new value minus old value must be equal to or greater than ten. This refers to the RC time constant, and if it increases by more than ten within 5 seconds the alarm will sound. The new value is updated in real-time while the old value will update itself to the new value every 5 seconds.

Another program was written using the threshold value as the alarm trigger instead. That means, instead of comparing new value and old value, it just reads the value in real-time and checks if it exceeds a certain number.

Slide 13:

Now that we have both programs up and running, how do we decide which program is better? We’ll need to develop a framework to rate how good each sensor is, and hence the SCAR criteria was born. It stands for Sensitivity, Consistency, Adaptability and Reliability, and I’ll be sharing what these are as well as how I measured them quantitatively.

Slide 14:

Firstly, sensitivity. This refers to the receptiveness of the sensor to an approaching hand, while consistency refers to the variation in data gathered from the previous test.

Slide 15:

Reliability is the frequency of false alarms, and is related to the consistency test, since both are evaluating how much the system can be trusted.

Lastly we have adaptability which is how adept the sensor is to changing environments and situations. We measure this by performing the previous three tests in different environments and then taking the average of all readings gathered.

The readings in each of the four criterion will then be converted into an arbitrary score out of 10.

Slide 16:

I then plotted a graph of the readings, and this brings us to the second last part, the Results & Discussion.

Slide 17:

To recap, method 1 used the threshold value and method 2 measured the increase within a period of time. As it can be seen from the graph, method 2 scored better in the SCAR criteria as compared to method 1. Method 1 is more sensitive because it used a cut-off point, but this also causes it to be very unreliable and unable to adapt to different situations due to the natural variation in the capacitance value of the aluminium foil.

Slide 18:

With this conclusion, we can move on to experiment 2 where we find the best value to use in measuring relative difference.

Slide 19:

Pictured is a graph showing how each of the SCAR criterion changes as the difference threshold value, or DTV for short, increases. As DTV increases, sensitivity as indicated by the blue line decreases while reliability increases. Adaptability was best at 18 DTV. Hence, 18 DTV is the best value to use as it offers the most adaptability, while lower values of 14-16 can also be used for greater sensitivity.

Slide 20:

In conclusion, we’ve covered the concept of capacitive sensing, the steps involved in capacitive sensing, a basic overview of the programmes written in the methodology and the SCAR framework used to determine alarm effectiveness. Through experimentation we found that measuring relative difference is more reliable and adaptable, and a DTV of 18 is the most adaptable.

So now I’ll be showing you guys a few pictures of the entire system in action.

Slide 21:

This is a photo of the set-up. Apart from the spaghetti wiring, you can see the distinct features of the alarm system, such as the aluminium foil, the LCD display, the speaker as well as some LEDs.

Slide 22:

When you hover your hand over the aluminium foil, this increases the RC time constant and sounds off the alarm. The speaker will start playing notes, the LED will light up and the LCD will change text.

Slide 23:

Finally, the LCD will change text again once the alarm stops sounding.

This concludes everything about my project. I hope you’ve learned one or two things about capacitive sensing and found it interesting. Capacitive sensing is actually used in a lot of things we see today, such as smartphone screens and non-mechanical buttons.

Slide 24:

To wrap up my presentation, I will be covering some general information about my time with research @ YDSP.

Slide 25:

I was attached to the Faculty of Engineering in NUS, and was mentored by associate professor Arun.

Slide 26:

My supposed attachment period with him was from 1st October to 31st December, daily from 9 to 3. But because this was a computing project, my actual attachment ended up being from 21st October to 31st December, where we met once a week to check progress. I worked the rest of the project from home, and this arrangement allowed me to commit to other stuff I had at that time.

Slide 27:

My project also undergone changes. My supposed project was originally to design a smart confined area with a security system, but we ended up changing our projects based on the areas we expressed interest in. I opted to go for capacitive sensing, while

Slide 28:

the other two people who were also working on the same project as me, but in different groups, changed theirs to the use of tilt sensors, lasers and gyroscopes respectively.

Slide 29:

So what’s the point I’m trying to bring across? The point is that R@YDSP is highly flexible, and that you don’t necessarily have to follow what the organisers originally intended. You can negotiate a schedule with your mentor as well as explore other areas of your project to work on.

If you’re interested in picking up some skills in science research, don't hesitate to sign up! It’ll be a fulfilling experience and will be something you won’t regret.

Slide 30:

This concludes my entire presentation. Thank you.