BeneSeat

Grace Ding
grace.ding1999@gmail.com

Jason Meyerberg
jason@footdoc.com

Catherine Huang
chuang@exeter.edu

Matthew Quan
quan.matthew1999@gmail.com

New Jersey Governor’s School of Engineering and Technology 2016

Abstract

As a result of the technological evolution in America, people are getting less and less exercise, resulting in an overwhelming number of adverse health effects. BeneSeat is a smart appliance aimed at those who sit for long periods of time and fail to take an adequate number of breaks. Through the use of a Particle Photon microcontroller and the If This Then That (IFTTT) framework to access the Internet of Things, users receive reminders to stand up and stretch after a period of time. Moreover, connecting to IFTTT and the Internet of Things allows BeneSeat to be more than just a health device; it allows BeneSeat users to automate their lives. In conclusion, BeneSeat is an extremely affordable, convenient and versatile device that improves users’ lives.

1. Introduction

Prolonged sitting has become ubiquitous in modern life. From transportation, to education, to the workplace, and the home, Americans sit an average of 13 hours a day [1]. Sitting has become such an epidemic that the term “sitting disease” has been coined to connote an increasingly sedentary lifestyle.

Sitting disease is a serious health concern and studies highlight the correlation between prolonged sitting and increased risk for various diseases. The primary goal of the BeneSeat smart cushion is to create a more effective and convenient solution to sitting disease by reminding users to periodically stand up and move around.

2. Background

2.1 Sitting Disease

Though it is not yet recognized by the medical community as a diagnosable disease, sitting disease has been linked to an increased chance for cardio-metabolic diseases such as obesity, type 2 diabetes, heightened blood pressure, heightened cholesterol levels, cardiovascular disease, and heart failure [2]. Sitting disease has become more prevalent as the proliferation of computers has made prolonged sitting a norm.

Currently, existing products, such as standing desks, attempt to alleviate sitting disease by encouraging people to stand while working instead of sitting. However, these products are expensive and require a full redesign of the desk [3].

In addition, studies have found that the only viable solution may be to break up these prolonged periods of sitting. A Canadian study found that people who exercised for up to an hour a day still exhibited symptoms of sitting disease. This suggests that exercise is not adequate to alleviate sitting disease. Other studies recommend that people move around one to
three minutes for every thirty minutes of sedentary behavior and a 2011 study even suggested that it is beneficial to both reduce time spent sitting and break up periods of sitting with breaks[4] [5].

2.2 Smart Appliance

A smart appliance is a household appliance (ie. washer, dryer, chair) that can access the internet and interact with other internet-connected devices, such as a smartphone, tablet, or other smart appliance. Smart appliances such as BeneSeat have the ability to connect to the Internet of Things to interact with other smart devices.

2.3 Internet of Things (IoT)

The Internet of Things is an idea in which everyday appliances are able to interact with one another via the internet to accomplish tasks. IoT devices help to reduce energy consumption, increase efficiency, and increase automation in everyday life [6]. Companies have created frameworks for the Internet of Things such as Apple’s Homekit, Google’s Project Brillo and Project Weave, and If This Then That. BeneSeat uses IFTTT to connect to the Internet of Things.

2.4 Particle Photon

The Particle Photon is a system created by Particle Industries Inc. that consists of the Particle Photon microcontroller, which is a programmable circuit board, and the Particle Build, which is a web based Integrated Development Environment. The Photon microcontroller has digital and analog input and output data pins. Digital pins handle a single byte of data while analog pins can send and receive 4096 bytes of data.

Users can program the Photon using the Integrated Development Environment, an application consisting of a code editor and compiler that allows programmers to write software, which is provided by Particle Industries. The Particle Photon uses the Wiring framework, an open-source programming protocol for electronics with a single microcontroller. Users are able to program the Photon using the languages of C or C++ [7]. To make BeneSeat a smart appliance, the Particle Photon is used to power and control BeneSeat.

2.5 Particle Cloud

The Photon is unique from other commercial microcontrollers because it has a built-in WiFi module that relays data to Particle Industries’ “Particle Cloud.” The Particle Cloud is a cloud based server that is able to connect to other IoT services such as If This Then That (IFTTT). Whenever the Photon sends data to other IoT services, it is first sent to the Particle Cloud, which then relays the data to the IoT service. The Particle Cloud is an integral part of connecting the Particle Photon to the Internet of Things [8].

2.6 If This Then That (IFTTT)

If This Then That is a free, web-based service that joins internet connected products. Chains of simple conditional statements or “recipes” allow over 300 integrated applications and products to interact with one another [9]. These conditional statements are composed in the format “if Event A occurs, then Event B should occur.” For example, a recipe BeneSeat utilizes is “if the Particle Photon detects a person sitting on BeneSeat, then silence an Android device.” IFTTT allows devices to connect to the Internet of Things because it provides a common framework for different devices to interact with one another.
3. Product Design and Testing

3.1 Cushion Design

During the design process, the main goal regarding the structure of the prototype was to maximize effectiveness and durability. To detect the presence of a sitting figure, button sensors are placed and wired inside the cushion. These buttons are sandwiched between two layers of rubber to maintain durability and comfort. Springs are added adjacent to the buttons in between the rubber layers to reduce the stress on the sensors and help the cushion retain its shape. As seen in Figure 1, memory foam acts as a barrier above and below the electronics to provide comfort. The entire product is encased in an 18 inch by 14 inch cushion cover to be able to fit on most conventional chairs.

18 mm tall buttons with large plastic caps were chosen over other sized buttons to reduce the overall height of the cushion while increasing sensor surface area to facilitate sitting detection. Both springs and wooden stoppers were considered to protect the button sensors. Springs were ultimately chosen because of their strength and ability to compress and decompress into their original shapes. In addition, elastic springs are less noticeable in the cushion than the rigid wood blocks. The buttons and springs are dispersed evenly in the electronics layer, as seen in Figure 2, to further reduce stress and maximize effectiveness.

To encase the layer of electronics and springs, rubber and plywood were both considered. Rubber was ultimately chosen, because it is more flexible than wood, allowing it to be less noticeable in the cushion. Lastly, rubber is much safer to use than plywood. While wood can crack under high pressure and burn if exposed to high voltage, rubber is more resilient and acts as an insulator to electricity.

3.2 Sensors

Buttons were chosen as the most appropriate sensor to detect sitting, because they are cheap and simple. Buttons return a single byte, zero or one, that corresponds to whether or not the button is pressed. Pressure sensors and temperature sensors were
Pressure sensors allow for more precise detections, but are too expensive. Temperature sensors can detect a rise in temperature in BeneSeat when a user sits on them, but they suffer from a lack of precision.

3.3 Circuitry

The buttons used in BeneSeat are four pin buttons. As seen in Figure 3, when the buttons are not pressed, zero voltage is going to the digital pin because the button switch is not completing the circuit between the 3.3 V output pin and the digital pin. When it is clicked, the button switch completes the circuit and the digital pin reads that a high voltage is being received.

The buttons are wired to the Photon in a parallel circuit. If the buttons were wired in series, all of the buttons would have to be pressed in order for BeneSeat to detect that a user is sitting. Wiring in parallel allows for detection of individual button presses which allows for much more precise sitting detection.

All nine buttons used in BeneSeat are connected in a setup similar to that seen in Figure 3. Unfortunately, the Photon only has two ground pins and one 3.3 V output pin. Therefore, there is one main ground wire that stems from the Photon and branches off into nine individual wires. These nine wires then connect to each of the buttons. A resistor is needed between the ground wire and the button so that the high voltage will not damage the ground pin. Similarly, there is only one 3.3 V output pin for the Photon, so the same branching technique is used to distribute power to each individual button.

BeneSeat utilizes nine buttons, but the Particle Photon has eight digital pins and eight analog pins. Therefore, eight buttons were connected to the digital pin and one was connected to an analog pin; a simple
conversion from analog to digital was used to read the one button connected to analog. The final wiring can be seen in Figure 4.

The Particle Photon has no external battery. When using the Photon, it must be plugged into a power source. For convenience, a Particle Photon Battery Shield and an 850 mAh Lithium-ion battery were installed into the Photon. To charge the battery, the user simply connects the charging cable to a power supply.

3.4 Electronics Case Design

A small plastic case houses and protects the Particle Photon, lithium polymer battery, battery shield, and charging cable. The case is custom-fit to the electronics to eliminate rattling and damage to components inside the case. The case is attached to the side of the cushion with VELCRO® fasteners. Designed in SOLIDWORKS, a 3D computer aided design software, and extruded in a 3D printer, the case measures 4.4 cm by 4.6 cm by 8.6 cm and is 0.2 cm thick. It is composed of three distinct parts: the main body (Figures 5 and 6), the top portion of the lid (Figure 7), and the bottom portion of the lid (Figure 8). The main body of the case consists of two shelves: a bottom shelf for the battery and a top shelf for the Particle Photon, battery shield, and charging cable. The battery is ventilated by slits on the back of the case to prevent overheating. The Particle Photon and battery shield sit on the top shelf. These shelves are depicted in Figure 5.

Because each half of the lid has a semicircular cut out, a hole is created in the center of the lid when both halves are placed together. This allows one side of the charging cable to remain outside of the case while the rest of the charging cable remains inside of the case. This design, shown in Figures 7 and 8, allows users to easily charge BeneSeat.
4. Code

4.1 Structure and Constants

The Particle Photon environment is coded in the C/C++ coding language using the Wiring framework. The structure of the code contains a setup() method that is called once in the beginning and a loop() method that is continually called.

In addition, the global variable NUMBER_OF_BUTTONS_PUSHED (NBP) stores the number of buttons currently being pushed, while the global variable STATE stores the sitting state of the user. STATE can be either STATE_SITTING, STATE_UNSURE, or STATE_STANDING; each is stored as an int. These two variables are instantiated as global variables so that they can be accessed in any part of the code.

4.2 Event Listeners

In programming, event listeners wait for an event to occur and execute pre-programmed methods when it occurs. Unfortunately, Particle does not provide any libraries for creating event listeners. To recreate event listeners that trigger when a button is clicked, the Photon must compare previous button states to the current button state. If the previous button state (clicked or unclicked) is no longer the same as the current button state, a sitting-detection algorithm is subsequently called. This Button-State-Changed (BSC) listener is implemented for all nine buttons. They work in conjunction to detect if a user is sitting on BeneSeat.

4.3 Threshold

In order to determine when a user is sitting, a definition of “sitting” in terms of button presses must be defined. Problems arise with an inaccurate threshold. If the threshold is too low, non-human objects can trigger BeneSeat, but if the threshold is too high, BeneSeat may not trigger when a user is actually sitting. An experiment was performed to determine the optimal threshold that would cause the BeneSeat to trigger the sitting protocols. While a sample size of eight may seem like a small sample group, the current available population is only 88. Therefore, eight is the highest sample size that passes the 10% assumption rule of inference of statistics, which states that the sample size must not exceed ten percent of the population size.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Minimum Buttons Pressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person 1</td>
<td>8</td>
</tr>
<tr>
<td>Person 2</td>
<td>7</td>
</tr>
<tr>
<td>Person 3</td>
<td>3</td>
</tr>
<tr>
<td>Person 4</td>
<td>5</td>
</tr>
<tr>
<td>Person 5</td>
<td>8</td>
</tr>
<tr>
<td>Person 6</td>
<td>3</td>
</tr>
<tr>
<td>Person 7</td>
<td>0</td>
</tr>
<tr>
<td>Person 8</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 1: Minimum buttons pressed by various subjects in product trial

<table>
<thead>
<tr>
<th>Object</th>
<th>Buttons detected (trial 1)</th>
<th>Buttons detected (trial 2)</th>
<th>Buttons detected (trial 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object 1 - Laptop (3.48 lbs)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Object 2 - Backpack (5.18 + .55 lbs)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Object 3 - Two dumbbells - Combined 60 lbs</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Object 4 - One dumbbell (30 lbs)</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2: Non-human object button press trials
In the experiment, a subject was asked to sit down on the BeneSeat for five minutes. During this time, subjects freely worked, drank coffee, or read. The minimum number of buttons was recorded, as seen in Table 1. This measurement is important because the threshold number should not be less than the minimum number of buttons pressed at any time. From this data, seen in Table 1, the threshold was determined to be three buttons pressed. Further analysis can be found in section 4.1.

In addition, to explore the possibility of preventing non-human objects from triggering BeneSeat, the number of buttons non-human objects triggered was recorded (Table 2). The weight of each object was also recorded in order to determine the weight at which objects pass the sitting threshold. A 0.55 lb wooden plank was placed on top of the BeneSeat to distribute the weight of the object. From Table 2, it is evident that a threshold of three buttons correctly disregards weights less than thirty pounds. Further analysis can be found in section 4.1.

### 4.4 Sitting-State-Change-Detection (SSCD) Algorithm

The BSC listener calls the Sitting-State-Change-Detection (SSCD) algorithm whenever it detects that a button has changed its state. NBP is updated with each trigger of the button listener. The SSDC then detects whether this change pushed NBP over the threshold, under the threshold, or did neither. Running this preliminary check eliminates any subsequent unnecessary method calls, saving computing and battery power.

The second part of the SSDC algorithm checks the current state. Subsequent code only runs if the current state is significant in relation to the change in number of buttons pushed. Table 3 is a state table that shows which combinations of state and change in NBP trigger a protocol. The elements in the table are the protocols BeneSeat will run if both conditions (row and column) are met.

### 4.5 Timers

The main function of the BeneSeat is to remind a person after a long sedentary period to stand up and take a break. Software timers are used to trigger this notification. When the SSDC algorithm detects a person in the sitting state, it begins a sitting timer that notifies the person to stand up when the timer completed. One complication is that people shift and adjust their position when they sit, possibly causing the premature release of all of the buttons. This would prematurely trigger the sitting timer to reset, because BeneSeat would think that the user stood up and would begin a new period of sitting. The third state, “Unsure,” solves this problem; essentially, if BeneSeat detects a person getting off of the seat before the sitting timer is completed, it begins an unsure timer. This timer runs for a

<table>
<thead>
<tr>
<th>Current State</th>
<th>Change in NUMBER_OF_BUTTONS_PUSHED (NBP)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NBP&gt;Threshold → NBP&gt;Threshold</td>
</tr>
<tr>
<td>Sitting</td>
<td>N/A</td>
</tr>
<tr>
<td>Unsure</td>
<td>Sitting</td>
</tr>
<tr>
<td>Standing</td>
<td>Sitting</td>
</tr>
</tbody>
</table>
minute. If the person sits back down within that minute, the unsure timer is expelled and the sitting timer resumes. However, if the person does not sit back down within that minute, BeneSeat assumes that the person went on a premature break, stops and expels the sitting timer, and runs its standing protocols.

4.6 Protocols

With each state, the BeneSeat runs different protocols. For each state, when the BeneSeat publishes to IFTTT that it is currently in the sitting state, IFTTT triggers applications or products that are set to perform an action when BeneSeat publishes that event. For example, when the BeneSeat notifies IFTTT that a user is sitting, IFTTT mutes the user’s phone and subsequently places the ringer volume to 100% when BeneSeat detects that the user has stood up. All of these protocols can be customized through IFTTT. Because IFTTT offers a vast number of integrations with a vast number of products, users have extreme customization with state protocols. In order to create a new IFTTT integration or “recipe” that will be added to a protocol, users simply need to log in to IFTTT and follow the seven step procedure.

5. Analysis and Results

5.1 Accuracy

In order to be a viable consumer product, BeneSeat must work accurately. Determining a correct button threshold is extremely important to the success of BeneSeat. As described in section 4.3, problems arise with an inaccurate threshold. An experiment to determine the optimal button threshold was conducted. Analyzing the data from this experiment, it was determined that the standard deviation for the minimum was 3.011, which is quite high. The subjects with minimums around seven and eight buttons were subjects who did not adjust their seat during the duration of the sit. Subjects with greater maximum to minimum fluctuations readjusted their seats during the duration of the test. These subjects were more valuable because they showed the minimum number of buttons pressed during a time when the user was just barely sitting.

The lowest minimum number of buttons pressed was zero. However, choosing zero as the threshold is illogical as stating that BeneSeat should perceive having zero buttons pushed as a user sitting makes no sense. Three is the next lowest number, with two subjects having three as a minimum. Three was chosen as the threshold number in order to encapsulate any occasion, including extremes, where a user is barely, but still sitting.

The non-human experiment reaffirmed three as the threshold number. A laptop, a filled backpack and a thirty pound dumbbell each pushed at most two buttons. However, a sixty pound dumbbell, which is the same weight as an eight year old child, pushed five buttons, as seen in Table 2[10]. Unfortunately, due to limited resources, more precise and variable weights could not be tested. But, it is clear that a threshold of three buttons will effectively prevent non-human objects from triggering BeneSeat. With the threshold set at three buttons, the BeneSeat activated 25 out of 25 times.

One issue that negatively affected the consistency of BeneSeat was the lag time between a state change and IFTTT triggering the corresponding protocol. In order to determine how large of an issue this is, the delays between a state change to sitting and to unsure were recorded, shown in Table 4. Both states were recorded to eliminate the confounding variable of different states being blocked. However, by comparing the means of the lag times of both states (9.714 seconds
for the sitting state, 8.714 seconds for getting up) using a two-sample mean t-test with a null hypothesis of the two means being insignificantly different and an alternative hypothesis of the two means being significantly different, the resulting p-value is .765, which is greater than an alpha level of .05. This means that there is 95% confidence that the null hypothesis should not be rejected and affirmed. Therefore, regardless of the state, IFTTT will have similar lag times.

The mean lag time for IFTTT is 9.6 seconds with a 2.6 second standard deviation (Table 5). While not egregious, a 9.6 second lag time can cause the user to become frustrated because it is easily observable. In section 4.5, the possibility of creating a custom application was mentioned. A custom application that communicates directly to BeneSeat from the connected device, rather than rerouting through IFTTT, would reduce lag time. A possible explanation for the lag time is the lack of a proper WiFi network.

Nonetheless, BeneSeat is extremely accurate and consistent with the detection of state changes, despite the lag between state changes and the IFTTT protocol calls.

5.2 Ease of Usage

It is imperative for BeneSeat to be easy to implement in consumer life in order to appeal to the mass markets. BeneSeat’s size mirrors that of other enterprise cushions and allows it to easily fit into any regular chair. Similarly, because BeneSeat is a cushion, there is no physical setup needed. Software setup is also easy because of BeneSeat’s integration with IFTTT. The user only needs to connect and download the IFTTT app and create recipes that are used with BeneSeat. BeneSeat is extremely easy to integrate and use in real life, making it more attractive to the consumer market.

5.3 Cost Analysis

The total cost of the BeneSeat components is $79.00. Key electronics such as the Particle Photon, battery shield, and battery cost $19.00, $13.00, and $10.00, respectively; other costs are broken down in Table 6. Though $79.00 may seem like a hefty price tag, it is important to note that the BeneSeat is in its prototype stage, so it is currently much more expensive than a mass-produced market version, because mass production will drive production costs down.
In addition, “normal” cushions can cost upward of $30.00, so a $79.00 price tag is not too excessive given the benefits of the electronic upgrade [11]. Furthermore, standing desks can cost hundreds and even thousands of dollars [12]. Keeping this in mind, the BeneSeat is a cost efficient alternative to other office health products.

5.4 Comparison with Other Products

There are many products that seek to counter the ever growing sitting disease. However, compared to more popular products available in the mass market, BeneSeat provides a more affordable and convenient alternative. BeneSeat is better than standing and treadmill desks, desks that are used at standing height to force a user to either stand or walk while working. These desks can cost hundreds and even thousands of dollars, making them unaffordable for many [12]. In addition, the benefits of BeneSeat are distinctly different from those of the standing and walking desk: to encourage people to burn calories by standing. BeneSeat, on the other hand, encourages people to take breaks from sedentary behavior in order to remain active. Standing desks and treadmill desks are also more difficult to setup as a user has to reset his/her current desk setup and accommodate for a higher desk setup, while setting up BeneSeat requires a user to simply put BeneSeat onto their chair.

5.5 Limitations

The current model of the BeneSeat is a prototype that was produced in a short period of four weeks, so there are many improvements that can be made. In the current model, BeneSeat connects to IFTTT. IFTTT allows BeneSeat to connect with over 300 applications and products. Although this provides the BeneSeat with versatility, it does not allow the BeneSeat to reach its full potential. Lack of time prevented the ability to create a BeneSeat smartphone application that would increase customizability of BeneSeat.

Lack of time and money also prevented an ability to experiment with other sensors and algorithms that could have led to a better sitting state detection algorithm.

The BeneSeat does not have a power button due to the design of the Particle Photon. Currently, the only on/off mechanism is unplugging the lithium ion battery that powers the Photon, which is extremely inconvenient. In addition, the current lithium ion battery is only 850 mAh, giving BeneSeat a short battery life. Concerns about short battery life would greatly discourage consumers from investing in BeneSeat.

Another limitation is the WiFi connection. BeneSeat was produced in an environment that did not provide an adequate and strong WiFi network. Therefore, the cushion had to be tested on an AT&T iPhone 6 WiFi hotspot. There was a visible lag time, as discussed in section 5.1. A lack of a variety of other wireless networks limited the ability to test whether or not this lag was due to the use of a hotspot to connect the Photon to IFTTT or the intrinsic integration and querying with IFTTT.

6. Conclusions

6.1 Summary

There is a growing amount of evidence in the scientific community that excessive sitting is significantly damaging one’s general health, leading to harmful health disorders and diseases. The BeneSeat encourages movement by reminding the user to take a break and reduce the amount of time spent sitting. Through the use of the Particle Photon and IFTTT, the BeneSeat is an affordable option for health monitoring and improving fitness. Hundreds of options on
IFTTT allow for custom “recipes” that open the door for endless possibilities in IoT technology. In general, the BeneSeat is a product that retrofits an everyday chair to relieve the effects of excessive sitting, improving health, one seat at a time.

6.2 Future Applications

The BeneSeat can become an even more powerful smart appliance. Originally, the BeneSeat aimed to improve health by reducing the amount of time sitting at desks, but future improvements can increase productivity and further automate life. For example, a smart light bulb in a room or desk lamp can be automatically turned on when BeneSeat detects a sitting state. Other examples of how BeneSeat can function with other IoT devices include changing room temperature, turning on a coffee maker, or ensuring that house doors are locked.

BeneSeat can also become smarter with improved hardware. Added lower back support can be utilized to improve detection of sitting states. In addition, this could be used to actively improve a user’s posture. Added lumbar support would allow BeneSeat to track and analyze user posture, aiding users in identifying poor posture. Additional sensors would also help improve the accuracy of BeneSeat. A vibration sensor could be used to detect when a person first sits down. Force sensors could be used to measure a weight sitting on the chair and very precisely determine if the user is sitting. Distance sensors can also be used to detect displacement of the top layer of rubber that occurs when a user sits down, which can improve sitting detection.

Improved sensors and improved algorithms can also lead to an increase in functionality, allowing BeneSeat to become a controller. One possible future integration for BeneSeat is to turn on a smart television when a person sits down. When BeneSeat detects a shift in weight to the left or right, BeneSeat can change television channels. In addition, a sequence of sit down, get up, and sit down can be used to snooze the sitting timer.

Analytical algorithms can also be provided with BeneSeat. As proven by the popularity of products such as the FitBit [13], health monitoring has become ubiquitous. BeneSeat has the potential to measure posture, calories burned, and time sitting. Being able to analyze and track personal health with the BeneSeat will improve its popularity in mass markets.

In the future, a BeneSeat app can be created to interact more directly with the product. Custom settings for the duration of the sitting and unsure timers and sensitivity of BeneSeat would give the user more control over the BeneSeat’s functionality. Similarly, a snooze function can be implemented. Alongside its IFTTT integration, a BeneSeat app would greatly expand BeneSeat’s functionality and appeal to the consumer market.

7. Acknowledgments

BeneSeat could not have been successful without the help of many people and organizations. Special thanks to the New Jersey Governor’s School of Engineering and Technology, without which this project would not have been possible. Thank you to Program Director Dr. Ilene Rosen and Associate Director Dean Jean Patrick Antoine for their leadership and support throughout the program. The BeneSeat team would also like to thank Lockheed Martin engineers Joshua Binder, Kyle Cavorley, Joe Ippolite, Joe Mirizio, and Shawn Vettom for mentoring and assisting with product design and testing.

In addition, many thanks to Residential Teaching Assistant Stephanie Tu for her advice and guidance throughout the course of this project, as well as Kristian Hu for also
assisting during the process. Finally, much gratitude to the rest of the Governor’s School staff and its sponsors, Rutgers, the State University of New Jersey, Rutgers School of Engineering, the State of New Jersey, Lockheed Martin, South Jersey Industries, Inc., and Printrbot, for providing this enriching opportunity.

8. References


Appendix A – BeneSeat Code

//this declares what each digital/analog pin is
//for example, set digital pin 0 to be the data for button 1
int button1 = D0;
int button2 = D1;
//This is repeated for all nine buttons.
int button9 = A0;

//these are the constants mentioned in section 4.1
int STATE;
int STATE_SITTING = 0;
int STATE_UNSURE = 1;
int STATE_STANDING = 2;

int NUMBER_OF_BUTTONS_PRESSED;

int button1SavedState;
//A saved state is created for all nine buttons.

//these are the timers mentioned in section 4.5
//the first argument is the duration of the timer; 1000 ms in a
//second, sixty seconds for a minute, x amount of minutes
//the second argument is the method to call when the timer completes
//the third argument is whether the timer is able to be repeated
//when it is completed
//unsureTimer should be able to repeat during one sitting, but
//sittingTimer does not need to automatically repeat because when
//the user gets up from sitting, that will automatically reset the timer.
Timer unsureTimer(1000*60*.5, runUnsureBreak, true);
Timer sittingTimer(1000*60*2.5, runSittingBreak, false);

//this is the button threshold, experimentally determined,
//mentioned in section 4.3
int THRESHOLD = 3;
//threshold is the number of buttons that needs to be pushed to
//be considered a "sit"

void setup() {

  STATE = STATE_STANDING;
  NUMBER_OF_BUTTONS_PRESSED = 0;

  //set each pin to an input mode
  pinMode(button1, INPUT_PULLUP);
  This is repeated for the other eight buttons.

  //all buttons are initially unpressed
  button1SavedState = LOW;
  This is repeated for the other eight buttons.
}

void loop() {

int receivedDataButton1 = digitalRead(button1);
// This is repeated for the other eight buttons, even button9, an analog connected button.
// By calling digitalWrite on an analog component, the Photon knows to convert this to a
// digital output.

// this is my Button event Listener for button 1 (mentioned in section 4.2)
// if the new state does not match my previous state, there was a state change
if (receivedDataButton1!=button1SavedState) {
    // All Serial methods are used for debugging
    Serial.print("Changed button 1! PrevState:");
    Serial.print(button1SavedState);
    Serial.print("changed to:");
    Serial.println(receivedDataButton1);

    Serial.print("number of button press");
    Serial.println(NUMBER_OF_BUTTONS_PUSHED);

    int prevNUMBER_OF_BUTTONS = NUMBER_OF_BUTTONS_PUSHED;
    if (receivedDataButton1==HIGH) {
        NUMBER_OF_BUTTONS_PUSHED++;
        Serial.println("the received data was high.
so before it was low and it changed to high.
so increase number of button push");
    }
    else {
        NUMBER_OF_BUTTONS_PUSHED--;
        Serial.println("the received data was not high
so it must've been low so decrease the number of button pushed");
    }
    // this is the logic seen in Table 3.
    if (prevNUMBER_OF_BUTTONS<Threshold) &&
    (NUMBER_OF_BUTTONS_PUSHED>=Threshold) {
        Serial.print("State");
        Serial.println(STATE);
        if (STATE!=STATE_SITTING) {
            Serial.print("changed from 1--->g and it was not sitting");
            runSitting();
        }
    }
}
else if (((prevNUMBER_OF_BUTTONS) >= THRESHOLD) &&
(NUMBER_OF_BUTTONS_PUSHED < THRESHOLD)) {
    if (STATE == STATE_SITTING) {
        Serial.print("changed from g-->l and it was sitting");
        startUnsureTimer();
    }
}
button1SavedState = receivedDataButton1;
// This is repeated for the other eight buttons.

void runSitting() {
    // called when the user sits down
    // Start sittingTimer and publish to IFTTT
    sittingTimer.start();
    Particle.publish("Sit","true");
    // if the unsureTimer was running because STATE = STATE_UNSURE,
    // stop the unsureTimer because the user is now back sitting
    unsureTimer.stop();
    Serial.println("running sitting, starting the sittingTimer");
}

void startUnsureTimer() {
    // start a timer. if during that time, you sit back down, delete the timer.
    // publish to ifttt
    unsureTimer.reset();
    STATE = STATE_UNSURE;
    unsureTimer.start();
    Serial.println("running unsure... unsure timer started");
    Particle.publish("Unsure","true");
}

void runSittingBreak() {
    // this runs when you have sat for one hour continuously
    sittingTimer.stop();
    STATE = STATE_STANDING;
    Serial.println("running break... sat too long");
    Particle.publish("Standing","true");
}

void runUnsureBreak() {
    // this runs when the user prematurely gets up
    // we publish a different state to ifttt because it is a different scenario
    // for example, a user would not want a notification telling him/her to get off
    // the seat when they already have!
    // (because this is the method that is called when they prematurely get up)
    STATE = STATE_STANDING;
    sittingTimer.stop();
    unsureTimer.stop();
    Serial.println("running break... got up already");
    Particle.publish("PrematureStand","true");
}
Appendix B – BeneSeat Recipes

If laser_chicken published Sit, then turn lights on

If laser_chicken published Standing, then set ringtone volume to 100%

If laser_chicken published Unsure, then send a notification

Figure 9: These are some IFTTT recipes that BeneSeat utilizes.
Images courtesy of IFTTT [14]