AN INTERNET-OF-THINGS APPLICATION TO ASSIST THE DETECTION OF FALLING TO THE GROUND

Yifei Yu[1], Yu Sun[2], Fangyan Zhang[3]

[1]Sage Hill School, Newport Coast, CA, 92657
[2]California State Polytechnic University, Pomona, CA, 91768
[3]ASML, San Jose, CA, 95131

ABSTRACT

As people get old, the risk of them falling increases; the fall will impact senior citizens more negatively than younger people. My grandmother once fell and hit her when she was alone at home, and she instantly became unconscious. Frequently, senior citizens are unable to help themselves after they fall, even if they remain conscious. However, there isn't a product that senior citizens can use to notify their relatives right away if they fall, and this leads to the question of how we can bring immediate aid to all senior citizens after they fall. This paper brings forward the product and software that can solve this problem. The product is a small wristband that detects any falls or collisions and notifies relatives right away. The software is an accompanying app that shows the data recorded from those falls or collisions, specifically designed for family members to keep track of their elders. We applied our application during our test sessions and conducted a qualitative evaluation of the approach. The results show that this experiment is a great solution to our problem, but with a few limitations and weaknesses.

KEYWORDS

Detection of falling, wristband, iOS, Android

1. INTRODUCTION

Falls are dangerous and costly. As people get old, the risk of falling increases; the fall will impact senior citizens more negatively than younger people. My grandmother once fell and hit her when she was alone at home, and she instantly became unconscious. No one in the family knew about this until someone arrived home and found that she fainted. This was a severe example, but senior citizens are unable to help themselves after they fall, even if they remain conscious. For instance, they might be unable to stand back up or call out loud. According to the Centers for Disease Control and Prevention (CDC), millions of older people—those 65 and older—fall every year, and one out of five falls causes a severe injury such as broken bones or a head injury [1]. Without immediate treatment of the patient, these injuries will only get worse and may even lead to death. For this reason, I created a small wristband that can detect an individual's falls and notify his or her relatives on the spot.

My mission is to help senior citizens receive the aid instantly after they fall, especially when no one is around. I designed a small, foldable wristband that can be worn around wrists or put into the pockets. It contains an accelerometer that detects a collision and records its magnitude, a GPS that marks the location, a clock that records the time of the fall, and a SIM card that can send these statistics to the phone and notify family members. Accompanying the wristband is an

David C. Wyld et al. (Eds): MLNLP, BDIoT, ITCCMA, CSITY, DTMN, AIFZ, SIGPRO - 2020
pp. 57-65, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.101206

application compatible with IOS and Android devices. It contains a built-in Google Map that shows the location of the last fall and its magnitude and time. Furthermore, aside from showing the latest data, it stores all of the past data and presents them on a Full Statistics screen. In the event of a fall, the accompanying app will notify the family members and show the fall data.

In this way, I strive to bring aid to senior citizens quickly after their accidents in the short run and aspire to reduce the number of unprotected accidents in the long term. My mission is to prevent senior citizens from being unattended after they fall by providing them immediate aid.

The market for fall detection and faster medical help is growing quickly due to the influx of new technologies [2]. Less than ten companies are developing or researching development of this kind of product (elderly fall sensors/help) [14][15]. According to some market research, I discover that some of the competing companies in the elderly care industry use techniques and systems that have been proposed to act as fall alert systems [3] [4]. These companies also sell fall alert systems to the elderly, in the form of bracelets that act as fall detectors. In the case of a fall, the bracelet contains a button that allows the user to call for help. Other companies offer their product with a similar function, but it involves the user physically typing a text message to the company's emergency help center to receive immediate aid. However, these proposals assume that the elderly are still conscious and able to move after they fall, which is rarely the case in practice. Furthermore, the products competing companies offer are not very user friendly due to their complicated user-interface, whether it is on the bracelet itself or on the accompanying phone application. The last problem is the battery life. According to my research, a company called LifeFone actually has very similar functions to the product I am developing [5]. However, their device (bracelet) only has a battery life of 5 days.

We have developed innovative measures to ensure the safety of senior citizens: an alert system that can bring aid to senior citizens in a quick and cost-saving way. The alert system has two components: hardware and software.

The goal of the application was to help senior citizens in case they fall and are unable to help themselves by immediately notifying family members. It should prevent the senior citizens from suffering too long from their accidents by bringing help to them right away. Requirements or Criteria included simplicity, user-friendliness in both APP and product (wristband), precision in location, magnitude, and time detected by the product (wristband) and its durability. The constraint included weak signal detection of the product (wristband) at some locations and its durability.

Our experiment includes the following materials: Particle Electron 3G-U260; Ublox SARAU260: GPS that pinpoints the location of the wristband with accelerometer attached to detect collisions or any forceful physical impacts; Google Maps: a map that shows the location of the wristband; Arduino: computing platform to program the wristband; and Thunkable: computing platform to code mobile APP[6] [7].

In our experiment, we first attached the Ublox SARA-U260 (GPS and accelerometer) onto the Particle Electron 3G-U260 (asset tracker), which made the hardware or wristband. Then, we linked the hardware to the Particle Console Web IDE to program. Afterwards, we coded the hardware in Arduino so that it showed the A, G, and Accel variables. The [A] variable shows a forceful physical impact upon the hardware. Such impact is above the safety threshold and would be considered a fall or collision, [G] variable shows the location of the fall. It provides the latitude and longitude of the GPS, and the [Accel] variable shows any physical impact that is below the safety threshold of the [A] value. Finally, we programmed the APP using Thunkable to show the location, time, and magnitude of the collisions. It includes a map (Google Maps) on

the top half of the screen and most recent fall statistics. The past falls are recorded into the cloud server.

Two experiments have been conducted to verify the following two aspects of the system. In Experiment 1, we tested the accuracy of the falling detection using different algorithms. The core falling algorithms rely on the accelerometers embedded in the system. We have tested the different machine learning algorithms to classify the falling status based on the sampled accelerometers values, such as SVM, RandomForest, and Decision Trees [9][10][11]. It turns out that RandomForest has the highest accuracy of 94.5%.

In Experiment 2, we tested the accuracy of the falling detection using different parameters. We tested the training dataset collected using different combinations of the accelerometer values such as (x), (y), (z), (x, y), (x,y,z). Although most of the accuracy is similar to each other, the 3value combination (x,y,z) is on average higher than the rest.

The rest of the paper is organized as follows: Section 2 gives the details on the challenges that we met during the experiment, including weak signals, the need to refresh data, and the lack of a button; Section 3 focuses on the details of our work including screenshots of the code and corresponding explanations; Section 4 presents the relevant details about the experiment we did, following by presenting the related work in Section 5. Finally, Section 6 gives the conclusion remarks, as well as pointing out the future work of this project.

2. CHALLENGES

2.1. Challenge 1: Weak Signal

Weak signals or connections have always been part of a challenge in this project. During testing sessions of the prototype (the bracelet), the data of the prototype sometimes doesn't show up on our computer screens. In addition, there have sometimes been flickering green lights on the prototype device, showing that our connection is weak. As a result, when we test the prototype by slamming it, the magnitude, location, and time of the collision doesn't show up on our computer screens, suggesting that the data is not received. This problem can be especially dangerous in the real world. If the user falls and no fall data is tracked, the user will not receive any aid and will suffer the consequences of the fall. Therefore, it is necessary for us to improve the signal and connection between the hardware (bracelet) and the API, which store and transfer the data, so that the high accuracy can protect our users.

2.2. Challenge 2: Need to Refresh Data

We test our prototype by slamming it against the table to resemble the user falling to the ground in real life. During our testing sessions, we come to realize that we cannot slam the prototype too often. In other words, if we slam it once at 1:00, we cannot slam it again 5 seconds later. That is because the accelerometer within the hardware is slow to record the data of the second fall. As a result, immediately slamming the prototype again after the first slam will not yield any data. What is even worse is that when we test it again minutes after the first test, the data still may not show up. As a result, we have to manually press the button on the side of the Particle Electron to refresh, and that's when it starts picking up data from our tests again. However, the user shouldn't have to press the refresh button at all in real life, because the device should be able to refresh itself and record all the necessary data.

2.3. Challenge 3: Lack of a Button

Our prototype, most of the time, is able to record the relevant data of the fall, including the magnitude, time, and location. Assuming that all of these features are recorded 100% of the time, the device still hasn't reached its full potential to help the user. It lacks a button. If the user hasn't fallen, but isn't feeling well, there should be a button he or she can press to immediately acquire help.

Standard A4 (210mm x 297mm) portrait page set-up should be used. The left, right, top and bottom margins should be 30mm. Do not use any headers, footers or footnotes. No page numbers. Single column. All main text paragraphs, including the abstract, must be fully (left and right) justified. All text, including title, authors, headings, captions and body, will be Times New Roman font.

3. SOLUTION

The alert system has two components: hardware and software. The hardware is a foldable wristband that can be worn on the wrist or put into the pocket. Inside, there is an accelerometer that detects a collision and records its magnitude, a GPS that marks the location of the fall, a clock that records the time of the fall, and a transmitter that can send these statistics to the phone and notify family members via the app. A safety threshold is programmed into the accelerometer to ensure that not every movement is considered a collision. The safety threshold, in other words, is an experimentally-tested number (16,000) that distinguishes the difference between a regular movement and an accident. Any recorded magnitude below the threshold (15,700, for example) is considered safe, and the device will report it to the app. On the contrary, any recorded magnitude above it (18,000, for example) is a fall, and the device will record its data along with the other relevant information. The corresponding phone app will notify family members immediately.

The software of the system is the accompanying app, a free download application that is compatible with IOS and Android devices. Family members can download it to track the elderly's locations and falls. Opening its home screen, the top half of the user interface is a built-in Google Map that shows the location of the latest fall and its magnitude and time. Aside from illustrating the latest data, it stores the past data on a Full Statistics screen.

The hardware and software work together to create an efficient system. During a collision, the hardware detects all the information and sends it to the app. The app will send notifications to the family members' phones through vibration and on-screen texts. We will sell our alert systems through online sales from selling platforms, and through direct sales with senior care centers. We believe that developing a cheap and efficient alert system will minimize the dangerous implications of a serious fall.

In the case of a fall, the bracelet will detect the location, time, and magnitude of the fall. All three pieces of the data will be transferred onto the app, which notifies family members and shows all the data transferred. This app has already been published and families can download it from both iOS and Android stores to keep track of their elders. Looking at the app, the top-half of the screen is a built-in Google Map which shows the location, bottom-left corner is the time, and bottom-right corner is the magnitude. For our purposes, we designed the magnitude from a scale from 1-5. 1 being the lightest fall and 5 being the heaviest.

Because we initialized accelThreshold, we can use it as a reference for reading accelerometer data. We set “if” statement at the beginning so that the device will only publish data if t.readXYZmagnitude() is greater than our defined safety threshold of 16,000. Any value greater than that threshold will show, because it is considered a fall; any value less than the threshold will not show, because they are regular movements the user makes, such as walking or running, which report a safe magnitude. The “if” statement that follows checks if transmittingData is true, and then publishes using the Particle.publish(“<readable name>”, <value>, 60, PRIVATE) function to output the value given a readable name. Then, the “if” statements underneath check if the GPS has a fixed point, and will proceed to transmit data if the fix is confirmed. Upon transmitting, the process of token delimiting occurs inside of the “for” loop and “while” loop and separates the Latitude and Longitude into two separate values for the gps_coords[] array.
Particle.publish(“<readable name>”, <value>, 60, PRIVATE) is run once again to publish the separated gps values.

```
void loop() {

    if (t.readXYZmagnitude() > accelThreshold) {
        // Create a nice string with commas between x,y,z
        String pubAccel_x = String::format("%d", t.readX());
        String pubAccel_y = String::format("%d", t.readY());
        String pubAccel_z = String::format("%d",t.readZ());
        String A_mag= String::format("%d", t.readXYZmagnitude());

        // Send that acceleration to the serial port where it can be read by USB
        Serial.println(pubAccel_x);
        Serial.println(pubAccel_y);
        Serial.println(pubAccel_z);
        Serial.println(t.readXYZmagnitude());

            // If it's set to transmit AND it's been at least delayMinutes since the last one...
        if (transmittingData) {
            lastPublish = millis();
            Particle.publish("A_mag", A_mag, 60, PRIVATE);
            Serial.println(A_mag);
            }
    }

    t.updateGPS();


    // GPS requires a "fix" on the satellites to give good data,
    // so we should only publish data if there's a fix
    if (t.gpsFix()) {
        // Only publish if we're in transmittingData mode 1;
        if (transmittingData) {
            // Short publish names save data!
            string = t.readLatLon();
            for(int j = 0; j < strlen(string); j++){
                LL[j] = string[j];
            }
            char * LatLon = strtok(LL, ",");
            int i = 0;
            while( LatLon != NULL ) {
                gps_coord[i] = LatLon;
                LatLon = strtok(NULL, delim);
                i++;
            }

            Particle.publish("G_lat", gps_coord[0], 60, PRIVATE);
            Particle.publish("G_lon", gps_coord[1], 60, PRIVATE);
        }
        // but always report the data over serial for local development
        //Serial.println(gps_coord[0]);
        //Serial.println(gps_coord[1]);
    }
```

Figure 1: key code in implementation

Figure 2: result from change in magnitude

The visual above depicts the change in magnitude with respect to time detected by the device during test falls.

Figure 3: result from change in latitude and longitude

The two visuals above show the change in latitude and longitude detected by the device in respect to time during a test fall. Both data are screenshots from ThingSpeak, our API.

4. EXPERIMENT

Our experiment includes the following materials: Particle Electron 3G-U260; Ublox SARAU260: GPS that pinpoints the location of the wristband with accelerometer attached to detect collisions or any forceful physical impacts; Google Maps: a map that shows the location of the wristband; Arduino: computing platform to program the wristband; and Thunkable: computing platform to code mobile APP.

In our experiment, we first attached the Ublox SARA-U260 (GPS and accelerometer) onto the Particle Electron 3G-U260 (asset tracker), which made the hardware or wristband. Then, we linked the hardware to the Particle Console Web IDE to program. Afterwards, we coded the hardware in Arduino so that it showed the A, G, and Accel variables. The [A] variable shows a forceful physical impact upon the hardware. Such impact is above the safety threshold and would be considered a fall or collision, [G] variable shows the location of the fall. It provides the latitude and longitude of the GPS, and the [Accel] variable shows any physical impact that is below the safety threshold of the [A] value. Finally, we programmed the APP using Thunkable to show the location, time, and magnitude of the collisions. It includes a map (Google Maps) on the top half of the screen and most recent fall statistics. The past falls are recorded into the cloud server.

Two experiments have been conducted to verify the following two aspects of the system. In Experiment 1, we tested the accuracy of the falling detection using different algorithms. The core falling algorithms rely on the accelerometers embedded in the system. We have tested the different machine learning algorithms to classify the falling status based on the sampled accelerometers values, such as SVM, RandomForest, and DecisionTrees. It turns out that RandomForest has the highest accuracy of 94.5%. In Experiment 2, we tested the accuracy of the falling detection using different parameters. We tested the training dataset collected using different combinations of the accelerometer values such as (x), (y), (z), (x, y), (x,y,z). Although most of the accuracy is similar to each other, the 3-value combination (x,y,z) is on average higher than the rest.

5. RELATED WORK

The first work is done by a company called Guardly [12]. It created a simple app that allows users to communicate whether or not they require assistance. They have also created a hardware device, but it has no fall detection. Our project differs from Guardly in that our app is more integrated to the users' needs. For example, our hardware alone is able to detect the fall of its users and automatically contacts family members for assistance. Furthermore, it automatically provides the fall statistics, including the location, time, and magnitude. Guardly, in contrast, requires users to manually type on the app to let people know that they need help. Although they have a good customer service center that receives users' call for assistance, their app adds inconvenience for the users.

Guardly is strong in that it has a clean-looking app. In other words, it has little to no bugs and sends users' message to their customer service every time. This is our weakness because sometimes our app glitches, so pressing a button may not do anything; the app will not receive the command or signal. Guardly is weak in that their concept is too simple: it contains only an app and a simple device. The device is almost useless as it performs the same function as the app, which is for users to call for help when they are not feeling well or when they fall. This field is our strength because our hardware can automatically detect users' fall and contact help (family members) for the users.

The second project/company we researched is Kitestring [8]. The company's concept is also very simple, as it sends automated SMS to the user's device (also a bracelet) and checks if they respond. If the user doesn't, it tells personalized emergency contacts. Our work is different from that of Kitestring in that we have both hardware and software, while Kitestring only has a hardware. Kitestring also requires users to manually respond to texts while our work does not require anything from the users. Compared with Kitestring, our strength is that we have both an app and hardware; it requires no customer service center which Kitestring has. In other words, Kitestring requires users to manually input a list of emergency contacts, and its customer service will send automated SMS to check on the user. If the user doesn't respond, it will notify the user's emergency contacts. It is so complicated. Our work, on the contrary, is very simplistic and user friendly.

Guardly's work has a portable fall detector that includes a GPS with Wifi, and can be worn as a necklace [13]. It also has a button that users can press in order to contact emergency help. Out of all works, Guardly has the most similar work to ours. However, one difference is that it also has a customer service center, which the device calls in case the user needs emergency assistance. For us, our device calls family members rather than the customer service. Since Guardly's work is very similar to ours, its strength is that it has a very strong and well functioning app and device. Because Guardly is already an established company, they do a very good job with minimizing glitches, so all data and messages are successfully transmitted between the user to

the API and to the customer service. This is our weakness because our system is full of glitches that need to be fixed in the meantime.

6. CONCLUSION AND FUTURE WORK

To solve the problem of senior citizens being unattended in the case of a fall, we developed a system that involved both hardware and software. The hardware was a bracelet for senior citizens that tracks the magnitude, time, and location of the fall. These data would be recorded on an API, which would also be transferred to the accompanying phone app that would show the data for family members. To test the accuracy of the alert system, especially the hardware, we ran two experiments. In the first experiment, we tested the accuracy of the falling detection using different algorithms SVM, RandomForest, and DecisionTrees. It turned out that RandomForest had the highest accuracy of 94.5%. In the second experiment, we tested the accuracy of the falling detection using different parameters. We tested the training dataset collected using different combinations of the accelerometer values. Although most of the accuracy is similar to each other, the 3-value combination is on average higher than the rest.

Currently, we haven't achieved the maximum amount of accuracy in our system due to weak connections between the device and API, and a weak refreshing system. Both of these limitations contribute to a less accurate system because data is not recorded at times.

The current practicality of the system is not very high due to the size of the prototype. Because the prototype is very big and bulky, it will definitely be too big for anyone to wear around their wrist once it's incorporated into a bracelet. Therefore, it is necessary to minimize the size of the physical components that record the data, so the bracelet will be small and convenient for all users. To maximize the usefulness the bracelet can bring for its users, a button must be added. Even if the user hasn't fallen, he or she can still press the help button on the bracelet if the user isn't feeling well, and will receive immediate help.

In the future, we plan to improve the accuracy of the system by boosting the signal connections between the device and API, strengthen the practicality by decreasing the size of the bracelet, and enhance the optimization by adding a "help" button. We strongly believe that by following these measures, our alert system will be able to reach its full potential.

REFERENCES

[1] Important Facts about Falls. 10 Feb. 2017, www.cdc.gov/homeandrecreationalsafety/falls/adultfalls.html. Lee, S.hyun. & Kim Mi Na, (2008) "This is my paper", ABC Transactions on ECE, Vol. 10, No. 5, pp120-122.
[2] Mubashir, Muhammad, Ling Shao, and Luke Seed. "A survey on fall detection: Principles and approaches." Neurocomputing 100 (2013): 144-152.
[3] Wu, Falin, et al. "Development of a wearable-sensor-based fall detection system." International journal of telemedicine and applications 2015 (2015).
[4] Mubashir, Muhammad, Ling Shao, and Luke Seed. "A survey on fall detection: Principles and approaches." Neurocomputing 100 (2013): 144-152.
[5] Matias, Igor, Nuno Pombo, and Nuno M. Garcia. "Towards a Fully Automated Bracelet for Health Emergency Solution." IoTBDS. 2018.
[6] Arduino, Store Arduino. "Arduino." Arduino LLC (2015).
[7] Levy, Paul Blain. "Thunkable implies central." (2020).
[8] Kitestring, www.kitestring.io/.
[9] Do, Thanh-Nghi, et al. "Classifying very-high-dimensional data with random forests of oblique decision trees." Advances in knowledge discovery and management. Springer, Berlin, Heidelberg, 2010. 39-55.

[10] Burrell, Jenna. "How the machine ‘thinks’: Understanding opacity in machine learning algorithms." Big Data & Society 3.1 (2016): 2053951715622512.
[11] Michie, Donald, David J. Spiegelhalter, and C. C. Taylor. "Machine learning." Neural and Statistical Classification 13.1994 (1994): 1-298.
[12] Beta, Brodie. “Guardly: An IOS App That May Save Your Life.” The Next Web, 7 Apr. 2011, thenextweb.com/apps/2011/04/08/guardly-an-ios-app-that-may-save-your-life/.
[13] IMANI, ANITA. "Design and development of a user interface for a mobile personal indoor navigation assistant for the elderly." (2014).
[14] El-Bendary, Nashwa& Tan, Qing & Pivot, Frederique & Lam, Anthony. (2013). Fall detection and prevention for the elderly: A review of trends and challenges. International Journal on Smart Sensing and Intelligent Systems. 6. 1230-1266. 10.21307/ijssis-2017-588.
[15] Chaudhuri, Shomir et al. “Fall detection devices and their use with older adults: a systematic review.” Journal of geriatric physical therapy (2001) vol. 37,4 (2014): 178-96. doi:10.1519/JPT.0b013e3182abe779