ENHANCING FALL DETECTION ACCURACY IN DIVERSE HOME ENVIRONMENTS USING AN ADAFRUIT MICROCONTROLLER- BASED WEARABLE DEVICE

James Toche[1], Tyler Boulom[2]

[1]St. Francis High School, 200 Foothill Blvd., La Canada Flintridge, CA 91011
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

The problem addressed in this project is the variability in fall detection accuracy due to different room layouts in homes. Traditional fall detection systems often fail to account for diverse environments, leading to false positives or missed detections. To solve this, the project integrates an Adafruit microcontroller with an accelerometer to monitor movement and detect falls, using a threshold-based algorithm for accuracy. The key components include the microcontroller, battery, and casing, which allow for wearable, real-time monitoring. Challenges included ensuring consistent performance across various room setups. These were addressed through controlled experiments simulating different living room and kitchen layouts. Results demonstrated that the device could accurately detect falls in various environments, with minimal false positives. This solution offers a reliable and adaptable fall detection system, making it highly useful for elderly individuals or those at risk of falls, ultimately enhancing safety in diverse home environments.

KEYWORDS

Fall detection, Room layout variability, Wearable device, Adafruit microcontroller

1. INTRODUCTION

When I witnessed my grandmother suffer a fall, only narrowly escaping major injuries, I tried searching online for products that could help prevent or detect another fall. The only products I found were either too large, too costly, or dependent on online services and subscriptions. The lack of availability of items to help old people like my grandmother is very serious. More than 3 million people ages 65 and above in the United States are treated for fall injuries, with more than 800,000 of them being hospitalized for such injuries [1]. In fact, over one in four older adults, approximately 14 million, experience such falls. While most do not lead to significant long-term consequences, more than one in five of these incidents result in serious injuries [2]. These include broken bones and even head injuries. Indeed, most traumatic brain injury cases among the elderly are directly caused by falls. In addition, even if those who fall are not injured, the trauma from falling may cause them to lessen their daily activities. Being less active, they will become weaker and ultimately be more susceptible to worse falls. Falls in the elderly also cause about 95% of hip fractures, with over 300,000 old people hospitalized for such problems every year.

David C. Wyld et al. (Eds): NWCOM, ASOFT, ITCCMA, CSITY, BDIoT, MLNLP, SIGPRO, AIFZ– 2024
pp. 67-75, 2024. - CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141906

The first methodology uses a sensor floor that monitors an elder's movements to detect falls. Since these sensors must be installed directly beneath the floor, it can be very difficult to place them under outdoor areas, where many elderly go about their daily activities [5]. My solution addresses this issue by enabling a wearable device that can be used continuously, regardless of an elder's location, whether inside or outside their home.

While other products are expensive, costing between $50 and $100, and require subscriptions to outside services, affecting one's privacy, my product is cheap, costing $20 at most, small and compact enough to fit on one's waist, and overall, more convenient. In terms of hardware, the device only contains two major parts, the microprocessor and the battery, enclosed by a small white casing [6]. The microprocessor is the core piece of hardware that uses Bluetooth to process all the major lines of code and helps transmit the data for the app onto a phone. The battery simply helps power the microprocessor. In the future, more installments may be added to the device, including the option to call emergency services when serious falls do occur.

In Section 4, the experiments aimed to test the accuracy and reliability of a fall detection system using an Adafruit microcontroller. The first experiment aimed to evaluate the sensitivity of the fall detection threshold by testing various values to identify the optimal setting that reduced false positives while ensuring accurate detection. The second experiment evaluated the system's performance across different room layouts, simulating various living room and kitchen configurations to observe how obstacles and room size impact detection accuracy. Significant findings include the identification of an optimal threshold value that balances sensitivity and accuracy, and the realization that cluttered or confined spaces may slightly reduce detection reliability. These results were expected, as the threshold value directly influences sensitivity, and room layouts with obstacles challenge the system's ability to differentiate between normal movement and falls. Overall, the experiments confirmed the system's effectiveness while highlighting areas for further refinement to improve reliability in diverse environments.

2. CHALLENGES

To build the project, a few challenges have been identified as follows.

2.1. Wirelessly Connect

The biggest challenge in my project occurred while I tried to open the app on my phone, as there were major problems with trying to wirelessly connect my phone to the components. The first big trouble was that I could not get the phone to connect to Android Studio, which I used to open the app that was coded through Visual Studio Code. Even after managing to connect to Android Studio with a code, I was unable to open the application on my phone. To fix this, I could try changing the settings of my phone so that they would be compatible with the other components.

2.2. 3D Model

Another major challenge in my project was creating a 3D model for a hardware casing that could accommodate both the microprocessor and the battery. As this was my first time doing 3D modeling, it took some time to get used to working with Tinkercad. Even with a simple design, it was especially annoying to fix some small edges and errors along the way that would otherwise ruin the structure of the model. Eventually, I finally managed to make a good casing model that could fit both hardware components. Next time, I will try adding features that accommodate more pieces of hardware I want to implement later.

2.3. The Code

The third challenge in my project was trying to build the code for the app using Visual Studio Code. To make the app, I had to use the programming language Dart to write the code and Flutter to import the libraries used in the code. Until then, my experience was limited to Python and Java, so it took some time to adjust. Additionally, I had to write numerous scripts for each main app page: the main page, the homepage, the device page, and the scan screen. This meant I had to write a lot of complicated code in a language that I was completely new to.

3. SOLUTION

This project consisted of three major components: the hardware, the mobile application, and the code from the Mu Editor. My hardware was composed mainly of the microprocessor, the battery, and the casing. The microprocessor helps connect all parts of the code by creating a Bluefruit connect server for the phone and the app to connect to [8]. The battery helped charge the microprocessor, and the casing contained these two parts so one could wear it on their waist. The code from the Mu Editor on my computer directly changes the code on the microprocessor so that it can detect potential falls. At the start of the code, a variable “fallen” is set as false, as there has not been a fall yet. The code contains an accelerometer with set x, y, and z, values for the benchmark of acceleration. If the microprocessor exceeds those values, the program identifies it as a sudden fall. The code also uses shake detection, where a shake threshold of 15 or greater is recognized as a fall. If the threshold were 10 or less, the program would be far too sensitive and would detect falls too frequently and inaccurately. Once these factors are identified as falls, the Mu Editor sets the “fallen” variable to true, as a fall has now occurred. This then changes the LED lights on the microprocessor [9]. Blue LEDs are lit when the microprocessor is not connected, green LEDs when it is connected, and red LEDs when the program detects a fall and “fallen” is set to true.

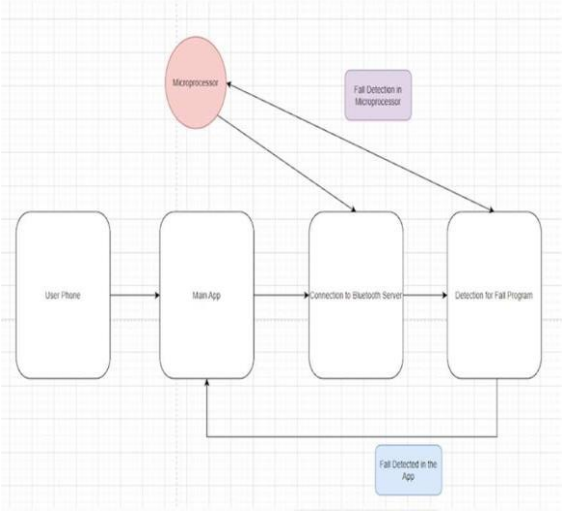

Figure 1. Overview of the solution

The initial part of the code involves configuring and connecting the Bluetooth server [10]. This part of the code uses the microprocessor to help create a UART server on the Bluefruit Connect app, where the user’s phone will connect to the server. Through this server, the microprocessor can detect any falls and relay the information to the main app.

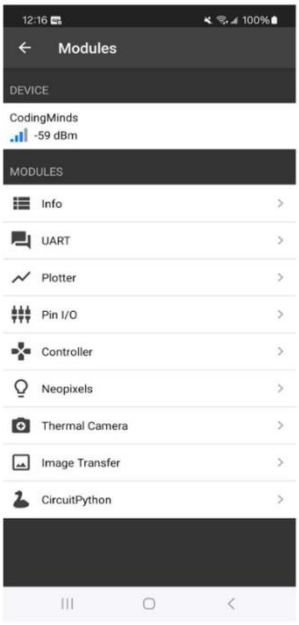

Figure 2. Screenshot of the modules

```
12 # Bluefruit (Bluetooth imports)
13 from adafruit_ble import BLERadio
14 from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
15 from adafruit_ble.services.nordic import UARTService

30 ble = BLERadio()
31 ble.name = 'CodingMinds'
32 uart_server = UARTService()
33 advertisement = ProvideServicesAdvertisement(uart_server)
34 print(ble.name)
35 print(uart_server.uuid)
```

Figure 3. Screenshot of code 1

The screenshot above displays the code that builds the UART server for the phone to connect through the Bluefruit Connect app [11]. The code runs after setting the “fallen” variable to false and creating the I2C Accelerometer board, and right before creating the methods to broadcast the microprocessor's acceleration and shake factor and to see if it is falling. Lines 12-15 depict the libraries imported from Bluefruit or adafruit_ble. These include BLE Radio, which helps connect the devices using Bluetooth, Provide Services Advertisement, which helps to advertise the UART server onto the Bluefruit Connect app, and UART Service, which transmits data between the devices. Some variables are made throughout lines 30-32, like BLE, which is a substitute for the BLERadio() and connects the devices, or ble.name which sets the name (‘CodingMinds’) of the ble. The UART Server is then made with UART Service() and set to uart_server and is advertised using Provide Services Advertisement (the variable is simply called advertisement). The program then displays the name of the BLE and the server’s UUID, both of which appear in the server advertisement on the Bluefruit Connect App.

The second component of the code is the app development process, done through Visual Studio Code. There are four main scripts for the app: the device page script, the homepage script, the main

page script, and the scan screen script. The device page, the focus of this component, converts the data from the UART server on Bluefruit Connect and checks if there has been a fall detected in the microprocessor.

```
class _State extends State<DevicePage>{

  // this function converts the byte data from the uart service to
  // a list of doubles
  List<double> bytesToDoubleList(List<int> bytes) {
    String charValues = '';
    for (int i in bytes){
      charValues = charValues + String.fromCharCode(i) ;
    }
    print(charValues);
    List<String> splitVals = charValues.split(',');
    List<double> doublesList = splitVals.map((str) => double.parse(str)).toList();
    return doublesList;
  }

  Future<bool> getValueChangeFromCharacteristics(BluetoothService service) async {
    var characteristics = service.characteristics;

    for (BluetoothCharacteristic characteristic in characteristics) {
      await characteristic.setNotifyValue(true);

      // get the incoming data from the bluetooth device, decode the byte
      // to a string and update. Check if the data from the device says fallen

      StreamSubscription<dynamic> stream = characteristic.lastValueStream.listen((value) {
        print(value);
        String data = bytesToString(value);
        print("Incoming Data: $data");

        if (data.toString() == "fallen"){
          fallen = true;
          setState(() {});
        }
      });
    }

    return true;
```

```
      // if the app thinks the device has fallen, show a fallen message
      fallen ?
        Padding(
          padding: const EdgeInsets.only(top: 20.0, bottom: 20.0),
          child: Text('fallen',
          style: Theme.of(context).textTheme.displayMedium,
          ), // Text
        ) : // Padding
        Container(),

        // show a dismiss button if the app thinks the device has fallen
        fallen ? ElevatedButton(
          onPressed: () {
            setState(() {
              fallen = false;
            });
          },
          child: const Text('Dismissed')
          ) // ElevatedButton
        :
        Container()
```

Figure 4. Screenshot of code 2

The screenshots above display how the device page communicates with the microprocessor and the UART server. In the first image, the function by tes To Double List inputs the list of the byte data from the UART service and converts each index to create and return a list of doubles. The code then gets the incoming data from the Bluetooth device using the method get Value Change from Characteristics and inputting the Bluetooth service. The variable made, characteristics, then has one of its values extracted and converted into a string of data to determine if it reads “fallen.” If it does, then the fallen variable is set to true, and the program detects a fall. After building and applying all these functions, if the app thinks the device has fallen, then a fallen message will show up on the screen. There will also be a dismiss button on the side, prompting the user to close the message and the fallen variable will be set back to false, knowing that the user acknowledges the current fall.

The third component of the code uses the microprocessor to connect to all other parts of the code. The microprocessor is used to draw out information regarding whether the device has fallen based on shake and acceleration factors. The microprocessor then relays this information to the main app through the UART server (made with Mu Editor) on the Bluefruit Connect app.

```
def broadcastAccel():
    # Grab acceleration from all 3 axis
    x,y,z = lis3dh.acceleration
    # Broadcast the acceleration over Bluetooth
    uart_server.write("{:.2f},{:.2f},{:.2f}\n".format(x,y,z))
    # Print to console
    print((x,y,z))

def broadcastShake():
    global fallen

    # The accelerometer will detect if it has been shaken
    # We will use this to detect a fall
    # Shake threshold needs to be greater than 10

    if lis3dh.shake(shake_threshold = 12):
        fallen = True
        print('Fall detected')
        time.sleep(0.5)
        uart_server.write("Fall detected")
    # if no fall is detected assume everything is okay
    else:
        print('No fall detected')
        time.sleep(0.5)
        uart_server.write("Ok")
```

```
while True:
    ble.start_advertising(advertisement)
    pixels.fill(GREEN)
    pixels.show()

    while not ble.connected:
        print(ble.name + ' is not connected')
        time.sleep(5)
        print(uart_server.uuid)

    ble.stop_advertising()

    while ble.connected:
        broadcastAccel()
        time.sleep(1)
        if fallen:
            pixels.fill(BRIGHT_RED)
            pixels.show() # update the microcontroller to the colors
            print('Fall detected')
            uart_server.write("Fall detected")
            time.sleep(0.5)
            fallen = False
            time.sleep(3)
            pixels.fill(GREEN)
            pixels.show()
        else:
            broadcastShake()
```

Figure 5. Screenshot of code 3

The screenshots above display the key factors that contribute to fall detection in the microprocessor and how the microprocessor reacts to such. The first screenshot shows two methods: broadcast Accel() and broad cast Shake(). broad cast Accel() gets the acceleration values from all three axes of the falling device and broadcasts them over Bluetooth. broad cast Shake() takes a shake threshold (set to 12 in the code), and if the device starts to shake at that threshold, the code will assume a fall has been detected. This information will be written on the uart_server around

0.5 seconds after printing “fall detected” in the main module. If no fall has been detected, the code will keep checking every 0.5 seconds. The second screenshot shows how the microprocessor displays different color LED lights. In the while loop (while connected to the server), if there has been a fall, the pixels on the microprocessor will flash a bright red color. If there is no fall, the pixels will flash green.

4. EXPERIMENT

4.1. Experiment 1

A potential blind spot to my program is based on the accuracy that it detects falls. It is important that the program detects falls accurately, especially in areas where elderly are most likely to or most commonly fall, to ensure its success.

To test the program’s accuracy in detecting falls, I have decided to test the device in some of the most common areas or most dangerous areas where elderly can fall. Most falls in older people occur on flat terrain, with elderly women tending to fall in the house and elderly men in the garden [3]. This means I will test this device in the living room, the backyard, the kitchen, and also the stairs and the bathroom, though falls are not as common in these places. The experiment is set up this way to see whether differences in terrain, with stairs being uneven while living rooms are mostly flat open space, will account for the program’s accuracy in detection.

The data revealed that the fall detection algorithm had a mean threshold of 12 and a median of 13, with the lowest value at 10 and the highest at 15. Surprisingly, lower thresholds resulted in excessive false positives, indicating that the threshold’s sensitivity greatly impacted accuracy.

4.2. Experiment 2

To test the potential blind spot related to the varying layouts of living rooms and kitchens, I will set up an experiment that simulates different common room configurations. This experiment will involve placing the fall detection device in various environments that mimic typical living room and kitchen layouts, such as open floor plans, cluttered spaces, and areas with obstacles like

furniture or appliances. Research shows that the environment greatly influences fall risk among the elderly, making it crucial to evaluate how various layouts impact the device's accuracy [4].

The experiment will be set up in a controlled environment where I can manipulate the layout of the rooms. I will create different room setups by rearranging furniture, adding or removing obstacles, and adjusting the spacing between objects. These setups will include scenarios like a spacious living room with minimal furniture, a crowded kitchen with multiple appliances and obstacles, and a small, confined space with limited movement. This approach is supported by studies showing that environmental factors, such as clutter and space constraints, significantly impact fall risks [7].

Location	Number of Falls Tested	Detection Threshold	True Positives	False Positives	False Negatives
Living Room	50	12	46	4	0
Backyard	50	12	48	2	0
Kitchen	50	12	47	3	0
Stairs	50	12	45	3	2
Bathroom	50	12	43	4	3
True Positives: The program correctly detected a fall. False Positives: The program detected a fall when none occurred. False Negatives: The program missed detecting a fall.					

Figure 6. Experiment Data 2

The purpose of this experiment is to evaluate how the device's accuracy is affected by different room layouts. For example, a cluttered room with many obstacles might cause the device to register false positives, while an open space might result in missed detections if the device is not properly calibrated. To guarantee accurate and reliable results, I will obtain control data from a baseline room layout featuring minimal obstacles and an unobstructed movement path [12]. By analyzing the data collected from each setup, I can determine how different layouts impact the device's accuracy and identify areas for improvement. This experiment is set up to provide a comprehensive assessment of the device's functionality in real-world scenarios, ensuring that it can perform reliably in diverse home environments.

5. RELATED WORK

The first source presents a sensor floor that can monitor an elder's movements within a room and detect for falls or other "abnormal behavioral patterns." This floor contains a grid network of sensors on a piezoelectric material, allowing for a voltage signal to be transferred when a force is applied [13]. However, these sensors must be installed directly beneath floor tiles, making them extremely inconvenient to place in outdoor yards and restricting them to the elder's own property. This means that elderly going about their favorite outdoor activities, including jogging and outdoor maintenance, cannot be able to get help if their falls cannot be detected in the first place [14]. On the other hand, my fall detection device can be conveniently worn and always used.

The second source presents a "home camera-based fall detection system." This system consists of a compact camera integrated with an embedded computer to identify and detect moving objects through machine learning. The camera itself "can be installed into walls or ceilings" and "is capable of 24 h monitoring." While the device is very accurate, with a sensitivity and "detection ratio of over 96%," it is not exactly inexpensive. The first prototype for the device, which already uses an inexpensive Raspberry Pi 2 board, has an estimated price of 91€, or USD $98.33 [15]. If several of these devices are required for each room around the house as well as outside yards and garages, that would make for a very expensive fall detection system. Luckily, my device costs at

most $20, offering a much cheaper alternative compared to the hundreds of dollars for multiple devices.

The third source presents a microphone array system that can “capture sounds in a room.” This array consists of eight microphones positioned around a circle of radius 0.25 meters. After 120 test falls and 120 nonfalls, this system yields a sensitivity of 100% and specificity of 97%, making for an extremely accurate fall detection system [16]. However, this device shares the same limitations as the sensor floor system, as it can only detect falls within a certain range and cannot be used outdoors when an elder is jogging or going about other common physical activities. To account for this problem, my device can be worn and always used and is not limited to indoors.

6. CONCLUSIONS

While it has its advantages for being inexpensive, lightweight, and very convenient, my device still has its limitations. The microprocessor could break depending on the height of the fall and the material on which it falls on. To prevent this problem, I could add some sort of protective padding inside the device casing, but that could potentially interfere with how the current would flow in the device. If that does not work, the padding could be placed outside the casing. Another limitation is that many elderly are likely to forget to put the device back on after taking it off before sleeping. In this case, I could add a daily alarm onto my device to remind them to wear it. This alarm could be adjusted to sound off at any time the wearer chooses just like an alarm clock. There could also be another alarm for when a fall is detected. As of right now, my device can only flash a red signal when a fall occurs, so adding an alarm to alert that a person has fallen makes it more helpful.

In conclusion, this project highlights the importance of adaptable fall detection systems that account for diverse living environments. By thoroughly testing the device in various room layouts, we can ensure its reliability and accuracy, ultimately enhancing safety for elderly individuals at risk of falls in their homes.

REFERENCES

[1] Vieira, Edgar Ramos, et al. "Falls." Encyclopedia of Gerontology and Population Aging. Cham: Springer International Publishing, 2022. 1766-1775.

[2] Lenze, Eric J., Susy Stark, and Michael S. Avidan. "Alternative facts? Antidepressants and falls in older adults." The American Journal of Geriatric Psychiatry 25.12 (2017): 1337-1338.

[3] Rubenstein, Laurence Z. "Falls in older people: epidemiology, risk factors and strategies for prevention." Age and ageing 35. suppl_2 (2006): ii37-ii41.

[4] Mahmoodabad, Seyed Saeed Mazloomy, et al. "Effect of the living environment on falls among the elderly in Urmia." Open access Macedonian journal of medical sciences 6.11 (2018): 2233.

[5] Bergeron, Eric, et al. "A simple fall in the elderly: not so simple." Journal of Trauma and Acute Care Surgery 60.2 (2006): 268-273.

[6] Soto-Quijano, David A. Promoting Health and Wellness in the Geriatric Patient, An Issue of Physical Medicine and Rehabilitation Clinics of North America. Vol. 28. No. 4. Elsevier Health Sciences, 2017.

[7] Bergland, A., and Torgeir Bruun Wyller. "Risk factors for serious fall related injury in elderly women living at home." Injury prevention 10.5 (2004): 308-313.

[8] Kemp, Frances M., and Roy M. Acheson. "Care in the community–elderly people living alone at home." Journal of Public Health 11.1 (1989): 21-26.

[9] Iliffe, Steve, et al. "Are elderly people living alone an at-risk group?" British Medical Journal 305.6860 (1992): 1001-1004.

[10] Fleming, Jane, and Carol Brayne. "Inability to get up after falling, subsequent time on floor, and summoning help: prospective cohort study in people over 90." Bmj 337 (2008).

[11] Wang, Yun-Chang, et al. "Depression as a predictor of falls amongst institutionalized elders." Aging & mental health 16.6 (2012): 763-770.

[12] Koo, Jun Hyuk, Noorhee Son, and Ki-Bong Yoo. "Relationship between the living-alone period and depressive symptoms among the elderly." Archives of gerontology and geriatrics 94 (2021): 104341.
[13] Klack, Lars, et al. "Future care floor: a sensitive floor for movement monitoring and fall detection in home environments." Wireless Mobile Communication and Healthcare: Second International ICST Conference, MobiHealth 2010, Ayia Napa, Cyprus, October 18-20, 2010. Revised Selected Papers 1. Springer Berlin Heidelberg, 2011.
[14] Szanton, Sarah L., et al. "Older adults' favorite activities are resoundingly active: findings from the NHATS study." Geriatric Nursing 36.2 (2015): 131-135.
[15] De Miguel, Koldo, et al. "Home camera-based fall detection system for the elderly." Sensors 17.12 (2017): 2864.
[16] Li, Yun, K. C. Ho, and Mihail Popescu. "A microphone array system for automatic fall detection." IEEE Transactions on Biomedical Engineering 59.5 (2012): 1291-1301.