Kaunas University of Technology

Faculty of Mechanical Engineering and Design

Modernization of Smart Knee Orthotic Device

Master's Final Degree Project

Shazia Khan

Project author

Assoc. Prof Inga Skiedraite

Supervisor

Kaunas, 2021

Kaunas University of Technology

Faculty of Mechanical Engineering and Design

Modernization of Smart Knee Orthotic Device

Master's Final Degree Project

Mechatronics (6211EX017)

Shazia Khan

Project author

Assoc. Prof Inga Skiedraite

Supervisor

Assoc. Prof. Saulius Diliūnas

Reviewer

Kaunas, 2021

Kaunas University of Technology

Faculty of Mechanical Engineering and Design

Shazia Khan

Modernization of Smart Knee Orthotic Device

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University.

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references.

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law.

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Shazia Khan

Confirmed electronically

Kaunas University of Technology

Faculty of Mechanical Engineering and Design

Task of the master's final degree project

Given to the student – Shazia Khan

1. Title of the project

Modernization of Smart Knee Orthotic Device

(In English)

Išmaniojo kelio sąnario atramos palaikymo mechanizmo modernizavimas

(In Lithuanian)

2. Aim and tasks of the project

Aim: Modernize smart knee brace and suggest running or walking speed and give alarm or notification when the user exceeds their safety threshold or limits.

Tasks:

1. Modernize smart knee brace to get real-time data such as knee angle, walking steps, pressure on the knee, temperature, live location, and walking speed.
2. Write program code for gyroscope sensor, pressure sensor, and GPS module sensor.
3. Store and analyze real-time data from the device in the cloud.
4. Create a mobile application to display real-time data acquired from the device and notifications.

3. Initial data of the project

Initial data of the project are gyro data x, y, and z, acceleration x, y, z data, and pressure sensor data in psi (Pound-force per square inch), GSM module data longitude, latitude and speed, and 3D model of knee brace.

4. Main requirements and conditions

Main requirements of the project are raspberry pi microcontroller, two gyroscope data initial position at 90-degree, pressure sensor data and gps module data.

Condition: gyroscope 1 should be place on thigh and gyroscope 2 should be place on shank and to know the live location GPS module should receive data from antena

Project author	Shazia Khan		
	(Name, Surname)	*(Signature)*	*(Date)*
Supervisor	Inga Skiedraite		
	(Name, Surname)	*(Signature)*	*(Date)*
Head of study field programs	Regita Bendikiene		
	(Name, Surname)	*(Signature)*	*(Date)*

Shazia Khan. Modernization of Smart Knee Orthotic Device. Master's Final Degree Project, supervisor Assoc. Prof Inga Skiedraite; Faculty of Mechanical Engineering and Design, Kaunas University of Technology.
Study field and area (Mechatronics): Production and Manufacturing Engineering (E10), Engineering Sciences (E).

Keywords: knee brace, knee angle, steps count, knee osteoarthiris.

Kaunas, 2021. Number of pages 56.

Summary

The review was done based on the article information of knee braces and latest technologies on which basis knee braces are working to better understood the working function of it. The technologies which have been depicted in this review are a piece of a more extensive image of improved brace treatment.

The modernization of smart knee brace based on hinged in knee brace and added gyroscope sensor, pressure sensor and GPS module on frame of knee brace. It made more advance knee brace where can be calculated the knee angle walking steps, pressure, speed, and show longitude and latitude of knee brace moreover. The python code is written to get real time data from the device and application programming interface (API) code to store real time data in cloud and share to mobile application to show real time data of knee brace in smartphone mobile application. Mobile application developed using React Native framework expo command and Java script. Primer data results gathered from the device to show accurate working results of the created model regarding equipment and programming qualities.

Khan Shazia. Išmaniojo kelio sąnario atramos palaikymo mechanizmo modernizavimas. Magistro baigiamasis projektas, doc. Inga Skiedraite; Kauno technologijos universitetas, Mechanikos inžinerijos ir dizaino fakultetas.

Studijų kryptis ir sritis (Mechatronika): Gamybos inžinerija (E10), Inžinerijos mokslai (E).

Reikšminiai žodžiai: kelio įtvaras, kelio kampas, žingsnių skaičiavimas, kelio sąnario artozė, knee osteoarthiris.

Kaunas, 2021. 56 p.

Santrauka

Apžvalga atlikta remiantis informacija straipsniuose apie kelio įtvarus ir naujausias technologijas, kad geriau suprasti jų veikimą. Šioje apžvalgoje nagrinėti kelio įtvarų technologijos su jau egzistuojančiais patobulinimais.

Išmaniojo kelio įtvaro modernizavime naudojami susilenkiantis kelio rėmas, giroskopo ir slėgio jutikliai bei GPS modulis, kurie tvirtinimi ant kelio rėmo. Tuo būdu padarė jį pažangesniu kelio įtvaru, kuris suteiki galimybę apskaičiuoti kelio kampo ėjimo žingsnius, slėgį, greitį ir be to, parodykite kelio įtvaro judėjimo ilgumą ir platumą. Phyton kodas parašytas, kad gautumėte realiu laiku duomenis iš įrenginių kelio įtvare ir programų programavimo sąsajos (API) kodą, kad saugoti įtvaro duomenis debesyje ir bendrinti juos mobiliojoje programoje, kad būtų matomi iš karto išmaniųjų telefonų programoje. Mobilioji programa sukurta naudojant „React Native framework expo “komandą ir „Java” scenarijų. Iš prietaiso surinkti pirminiai duomenų rezultatai parodo sukurto modelio tikslius veikimo rezultatus priklausančius nuo įrangos ir programavimo savybių.

Table of contents

List of figures

List of tables

Introduction

Smart knee orthotic devices are wearable device, and it deals with the inconvenience of knee osteoarthritis. It is a support that helps to reduce knee torment by moving the weight of the most harmed part of the knee and monitor your knee activities. A smart knee orthotic device is also known as a smart knee brace. In the present scenario smart knee brace comes with multifunction, it offers advanced features which help the user to see their daily activities and protect them from future injuries. Smart knee braces are equipped with sensors that use self-learning algorithms that can measure the load on the knee. These types of braces help osteoarthritis patients in the future to properly regulate their day-to-day physical activities. It helps doctors to see patient's progress for fast recovery.

Smart knee orthotic devices monitor and encourage patient's compliances, helps to prevent patella injuries, and helpful for Total Knee Arthroplasty (TKA) and application conjunction in the device allows clinician or physician to check and monitor patient's balance.

Aim

Modernize smart knee brace and suggest running or walking speed and give alarm or notification when the user exceeds their safety threshold or limits.

Tasks:

1. Modernize smart knee brace to get real-time data such as knee angle, walking steps, pressure on knee, temperature, live location, and speed.
2. Write program code for gyroscope sensor, pressure sensor and GPS module sensor.
3. Store and analyse real time data from the device in cloud.
4. Create mobile application to display real-time data acquired from the device and notifications.

1. Literature review of smart knee brace

Knee agony is an inexorably normal issue for youthful competitors and may result from both intense and abuse wounds. There are varieties of knee braces and it's important for doctors or health and cares professional to help to decide which knee brace might be appropriate for patients according to their knee injuries and torment. The specific knee support item you pick, however, will rely upon what sort of knee torment you are managing, why you are managing that torment, and what sort of exercises you are attempting to do while wearing the support, sleeves, and strap.

Knee braces need to wear in case you have knee agony or forestall injuries while playing giant corporeal sports wherever high chances of a knee injury. Knee brace could be utilized for rejuvenating motive, for illustration, ACL injury. The brace gives moderate, confined development permitting the victims to progressively recapture the inherent purview of gesticulation. Knee braces as well prove to be convenient for linkage swelling victims as they could be helpful to decreasing torment and aggravation. The physiotherapist may assist you with choosing a knee brace if you need to wear [1].

1.1. Types of knee braces

Generally, doctors suggest knee braces to patients according to the level of protection in knee braces, based on knee pain and prevention from injuries. There are three levels of protection in knee braces.

- The first level of brace provides a minimal measure of assistance, although this is the extreme pliable, for illustration, a knee sleeve. It is premier for relief from soreness and gentle to direct help while enduring thoroughly vital [2, 3].
- Second-level braces provide more security than the first level, they are not as versatile, yet take into consideration the scope of development. Wraparound braces and knee lashes are authentic mock-up. You could get gentle to direct knee uphold for relaxation from soreness related to tendon dangers and tendonitis [2, 3].
- The third level brace, for illustration, a hinged knee brace, provides the most advantageous however confined evolution. This variety of braces are as well often heavier. It's finest for recuperating from a medical course of action when knee development should be restricted to forestall re-harming individuality. To make it a stride further, there is consistently the choice of a 3+ Level for Maximum Protection. This level is righteous for helping in relief from soreness and backing for moderate to significant dangers and conditions [2, 3].

There are five kinds of knee braces usually doctors suggest to patients as respecting their torment.

1. Knee sleeves
2. Wraparound or Dual wrap braces
3. Hinged knee braces
4. Knee straps
5. Closed and open patella braces

1.1.1. Knee sleeves

It comes in several sizes, and you could slither them directly over the knee. It gives knee coercion, which aids to control expansion and torment. Knee Sleeves regularly function admirably for mellow

knee torment, and it is helpful for arthropathy pain. Sleeves are pleasant and could be suited under the dress [4, 5].

Fig. 1. Knee sleeve [5]

Advantages:

- Legitimate fitting pressure pieces of clothing increment blood dissemination and help to lessen the development of lactic corrosive in the muscles. A sound blood and oxygen stream better help quicker muscle recuperation or recovery [6].
- Support your muscles during difficult exercise/exercises and lessening the measure of solid vibrations, decline muscle exhaustion, and improves athletic perseverance [6].

Disadvantages:

- It's not to be utilized as a replacement for massage, extending, and legitimate rest and care for muscle or potentially joint wounds or injuries [6].
- Not all body types are equivalent, and the ideal weight focuses for the pressure pieces of clothing may not function admirably with your shape. Indeed, even with the various sizes gave, in some cases, they simply don't function admirably [6].

1.1.2. Wraparound or Dual wrap braces

It functions effectively for a sports person encountering mellow to direct knee agony, it is advantageous than sleeves. This type of supports is uncomplicated to wear and remove and could be utilized while training, it does not have the mass and weight of hinged supports [5, 6].

Advantages:

- It permits more weight to be lifted in the squat.
- It reduces stress and pulling forces on the quadricep tendon.
- Decreasing the weight on the ligament assists with trying not to isolate your ligament from the patella.

Disadvantages:

- It can build friction between the patella and the ligament of the kneecap; this can prompt injury and inconvenient knee joint issues like joint inflammation.

– It changes the muscles focused by the activity expanded erosion in the knee joint.

Fig. 2. Wrapearound or Dual wrap braces [6]

1.1.3. Hinged knee braces

It is regularly utilized autopsy, for outpatients and sportsperson who needs an extra significant level of indemnity and backing. This class of caliper (Fig.3) retains your knee in the finest contortion when it twists, to help recuperate and maintain a strategic distance from the lesion. A primary care physician may suggest hinged knee support after abscission, yet another kind of support when you have arrived at a definite end in the recuperating cycle. Pivoted supports are also stiff or slushy, with slushy ones offering discourteous than stiff supports or rigid braces [7, 8].

Fig. 3. Hinged Knee Brace [8]

Advantages:

– It gives a greatest degree of help to the knee.

- Ideal for everyday exercises and all games for the individuals who are experiencing mellow to direct tendon wounds or insecurities, meniscus wounds, injuries, or osteoarthritis.
- It retains your knee in the finest contortion when it twists, to help recuperate and maintain a strategic distance from lesion [7].

Disadvantages:

- On the off chance that the brace is excessively tight, it might cut off flow or squeeze a nerve.
- Wearing a tight brace for a really long time may bring about additional issues other than straightforward distress [7].

1.1.4. Knee straps

It is the perfect infusion in the event that you experience affliction on knee by reason of runner's knee or jumper's knee (Patellar Tendonitis), Osgood-Schlatter Disease, or Patella Tracking. It can fit under the clothes and yet hard to wear and eliminate.Wearing this sort of lash forestalls patella wounds and limits knee torment by placing pressure on your Patellar Tendon (Fig. 4) [9].

Fig. 4. Knee strap [9]

Advantages:

- It helps to relieve pressure off the patellar ligament in the knee.
- It gives some insurance and extra dependability to the knee after it has been harmed.
- Intended to forestall injury to the knees during physical games.

Disadvantages:

- At the point when the patellar ligament is stressed past its typical scope of movement, it can cause irritation and patellar tendonitis - prompting knee torment and inconvenience.

1.1.5. Closed and open patella braces

Closed and open patella the braces look the same, but the open patella has a hole in the middle place where there is no hole in the closed patella brace. A closed patella brace provide compulsion at the kneecap with similar compulsion as the remainder of the knee and extra help and open patella brace

permit alleviation of knee compulsion and further kneecap uphold with legitimate gesture and tracing (Fig. 5) [10]

Fig. 5. (a) Closed patella brace (b) Open patella brace [10]

Advantages:

- It offers help for the knee joint just as different advantages, for example, pressure, heat maintenance and insurance from additional wounds.
- This assists with forestalling unusual following of the patella during exercises, ensuring that it remains in appropriate arrangement with the remainder of the joint.
- It likewise assists with soothing tension on the patella.

Disadvantages:

- It can cause skin irritation because of full coverage.
- It changes the muscles focused by the activity expanded erosion in the knee joint.

1.2. Modern technologies added in knee braces

Now a day braces are available with more advanced technologies, which help patient and doctor to track their daily life activities while wearing the device, for example, a device connected with a mobile application shows how many steps a person walked [11]. Likewise, DJO company developed smart knee brace with motion intelligence [12] and EOS and Blockbox solution developed smart knee prosthetic containing embedded sensor and connectivity which allows doctors to get real-time data, potentially speeding patient recovery [13].

1.2.1. X4 smart knee brace with motion intelligence

This is a wraparound knee brace with a cloud-based remote patient monitoring solution. X4 with MI underpins the knee while permitting medical services experts to assess knee joint scope of movement, at-home exercise consistency, and changes in torment level. This is the first knee brace (Fig. 6) for the complete knee arthroplasty market that screens continuously and empowers network all through the continuum of care [12].

Fig. 6. X4 smart knee brace [12]

1.2.2. Smart customizable knee orthosis

It has smart knee joint monitoring. It is a combination of embedded sensors and connectivity that allows doctors to analyze data in real-time. This device (Fig. 7) is helpful for torn ligament which caused by wrong movements or exercise and extreme actual tension on the knee, the most well-known knee wounds and ordinarily require a long recuperation period with a danger of long-haul incapacities. Recuperation periods differ intensely dependent on the patient's development conduct. Putting too substantial a heap on the knee too soon can delay recuperation, as cannot moving regularly enough. Be that as it may, most patients don't have the foggiest idea of what's to an extreme and what's insufficient. Criticism from a shrewd gadget could forestall overexertion or unnecessary limitations [13].

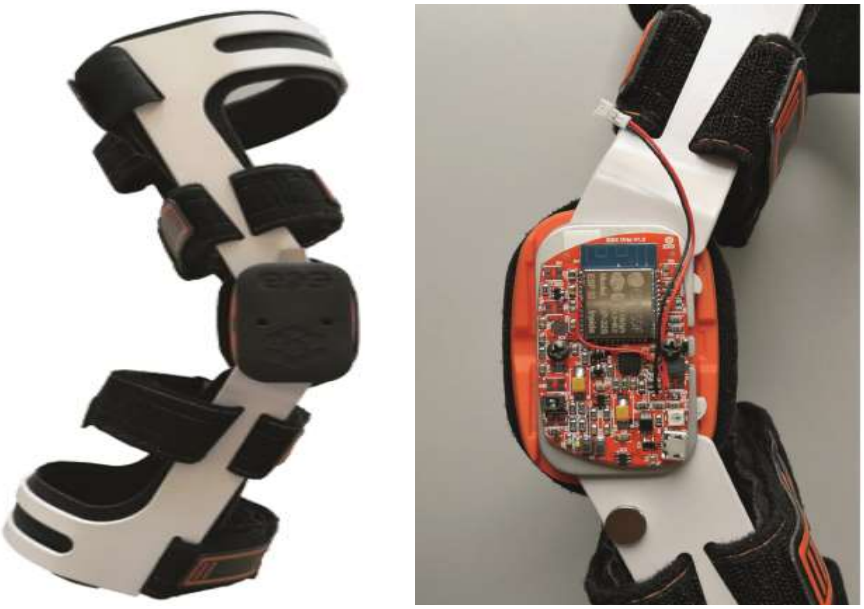


Fig. 7. Smart customizable knee brace by EOS and Blockbox Solution [13]

2. Design of the device and hardwares

As per the aim, modernizing the smart knee brace and adding features which will be helpful for patients for fast recovery especially for total knee replacement rehabilitation and knee osteoarthritis patients. For modernization of smart knee brace, designed the knee brace which has similar design with the hinged knee brace. In (Fig. 8**)** shows the 3D model of a device, designed in SolidWorks. The 3D model of a prototype designed according to tasks of the project and in (Fig. 9**)** shows the name of all sensors and devices added to the device.

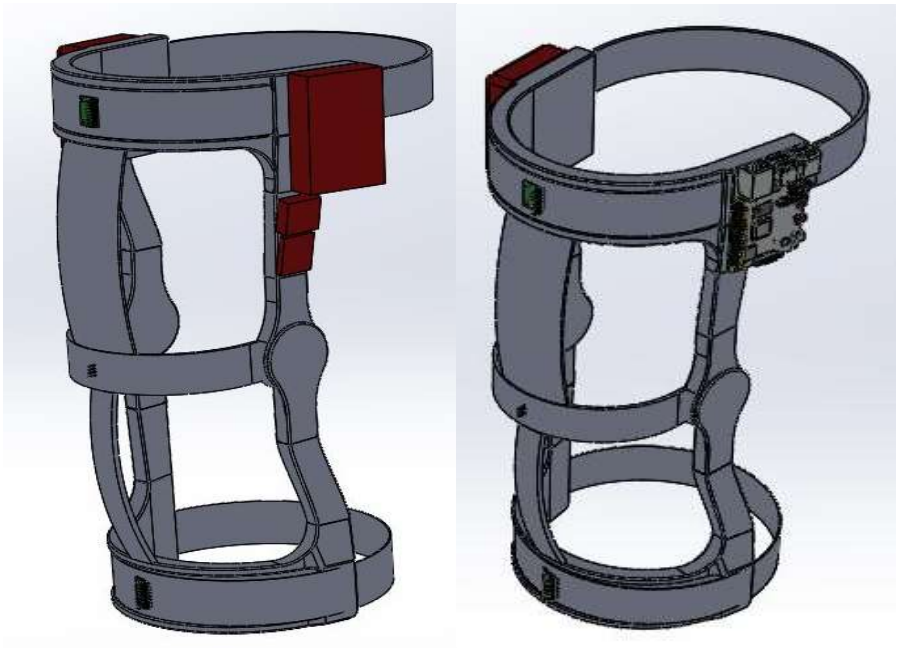


Fig. 8. 3D model of Protype device

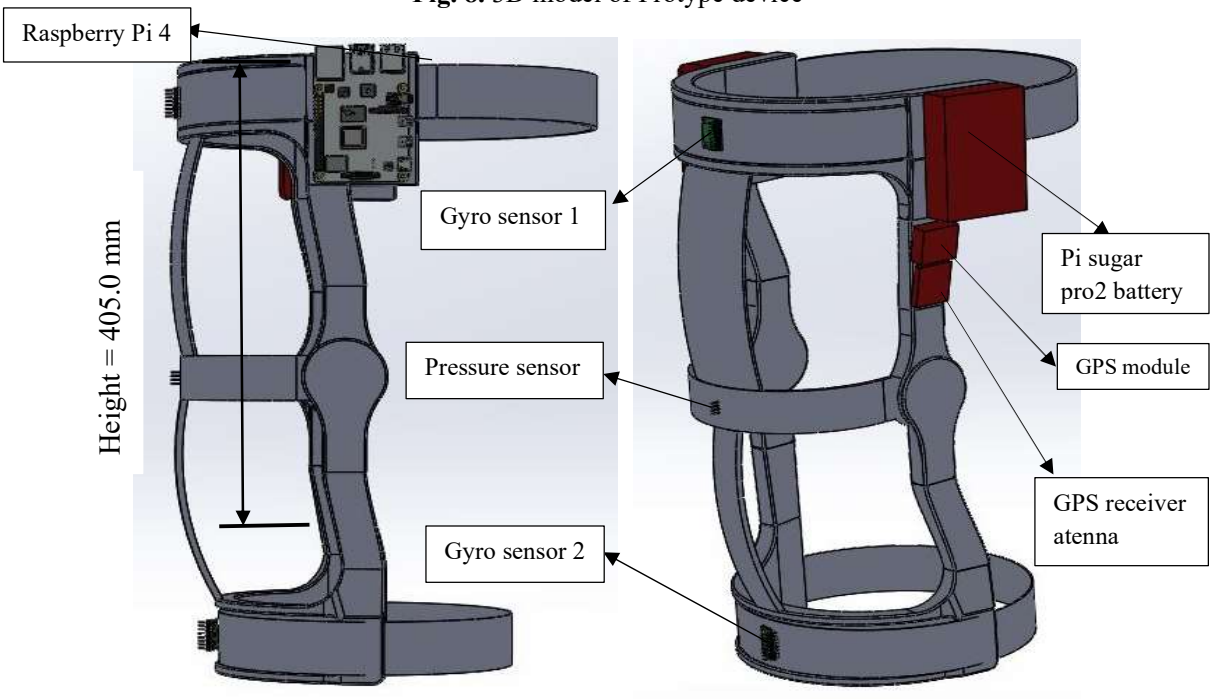


Fig. 9. Name of the sensors and devices attached in knee brace.

2.1. Requirement for device

To design the device, microcontroller, sensors, jumper wires and power supply for device are required

- Raspberry pi 4B microcontroller
- Pi sugar pro 2 battery
- Gyroscope sensors MPU6050 (2 pc)
- Pressure sensor BMP180
- NEO 6MV2gps module sensor

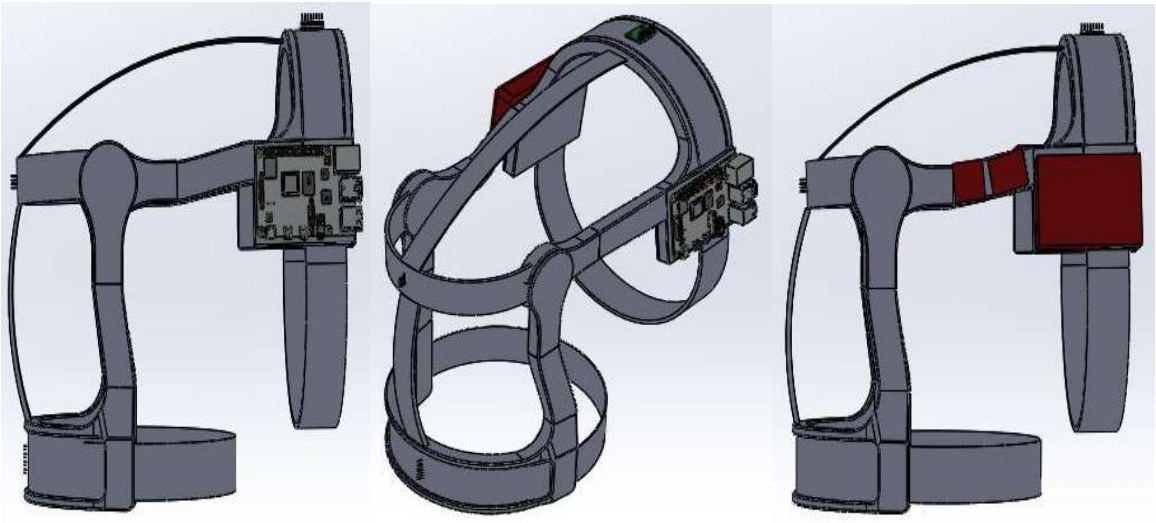


Fig. 10. Different orientations of 3D prototype of smart knee brace

In (Fig. 10) shows the different orientations of smart knee brace and represent how much knee brace can bend and calculate knee angle.

2.2. Data requirement for device

According to the undertaking, calculating knee angles, walking steps, walking speed, live location, and pressure on knee and temperature. To make the knee brace more advance utilizing the microcontroller, sensors, underneath (Table 1) appearance rundown of the sensors, area and uses.

Table 1. Sensor names, location, data and uses of the sensors.

Sensor name	Location of sensor	Data from sensor	Uses
MPU6050 Gyroscope sensor 1	Thigh	Gyroscope data: gyro x1, gyro y1 and gyro z1 Acceleration data: accel x, accel y and accel z	To calculate the knee angle and count walking steps
MPU6050 Gyroscope sensor 2	Shank	Gyroscope gyro x2, gyro y2 and gyro z2	To calculate the knee angle
BMP180 pressure sensor	On knee	Pressure in psi	To check pressure on knee
NEO 6MV2 GPS module	Right corner of frame	Latitude, Longitude and Speed	To generate live location and speed.

2.3. Specifications of microcontroller and sensors

Gadgets frequently use various microcontrollers that cooperate inside the gadget to deal with their separate undertakings. In every device, there is one or more controller which controls the device and store data from the other device and sensor. Microcontrollers are utilized in a wide exhibit of frameworks and gadgets. A sensor is a gadget that actions actual contribution from its current circumstance and converts it into information that can be deciphered by either a human or a machine.

2.3.1. Raspberry Pi 4 B microcontroller

In this device raspberry pi, 4 model B microcontroller is used. Raspberry Pi 4 Model B is the latest thing in the renowned Raspberry Pi extent of PCs. It offers profound rates up, blended media execution, memory, and accessibility stood out from the prior age Raspberry Pi 3 Model B+, while holding reverse similitude and similar force use. Raspberry Pi 4 Model B gives work territory execution basically indistinguishable from entry level x86 PC systems for the end customer. In (Fig. 11) shown the raspberry Pi 4 and in (Fig. 12) the name and number of all I2C pins of Raspberry Pi 4

Fig. 11. Raspberry Pi 4 model B [14]

The motivation to pick the raspberry pi 4B microcontroller is it is on various occasions faster than an Arduino with respect to clock speed. Altogether more evidently censuring for Arduino, Pi has on numerous occasions more RAM. The Raspberry Pi is a free PC that can run a real working structure in Linux.

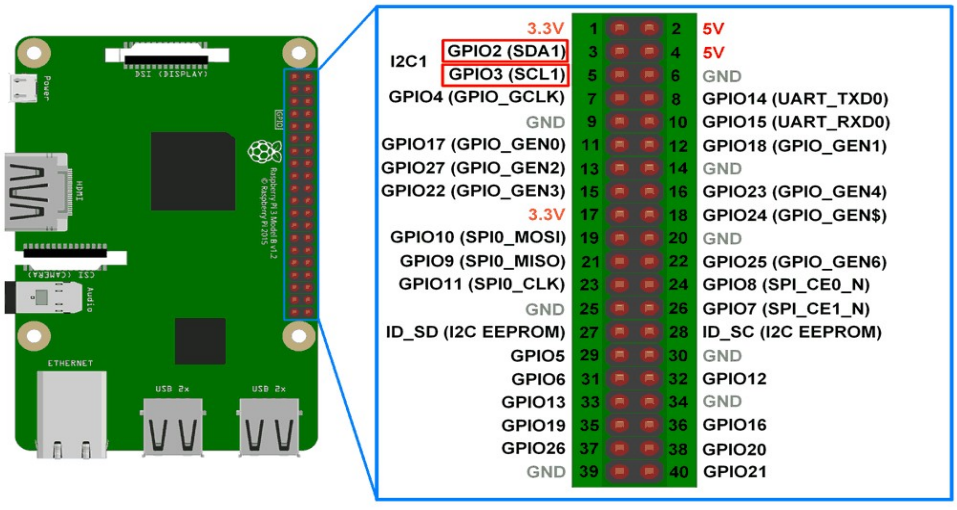


Fig. 12. I2C pin configuration of raspberry pi 4 [15]

2.3.2. Pisugar 2 Pro Battery

Pisugar Pro 2 is a portable power battery specially designed for the raspberry pi microcontroller, it does not require soldering, utilizes exceptional pogo-pin to supply power, doesn't possess any GPIO of Raspberry Pi, can be utilized on Pi both with or without a GPIO header. It has a 5000mAh battery capacity, it gives 8-10 hours battery life, a USB charging port, micro-USB, and type C. In (Fig. 13) showing Pisugar Pro2 battery.

Fig. 13. Pisugar Pro2 Battery [16]

2.3.3. MPU6050 Gyroscope sensor

MPU6050 is a MEMS device with six degrees of freedom inertial measurement unit sensor and IMU sensor in short, so the 6-axis consists of a 3-axis accelerometer and 3-axis gyroscope. In addition to these components, we have an invert temperature sensor and digital motion processor which makes complex calculations on itself and makes the work of the microcontroller much easier. The MPU6050 is generally accompanied by model number GY-521. In (Fig. 14) showing MPU6050 gyroscope sensor.

MPU6050 is cheap in cost, small, and easily available. It has an auxiliary I2C bus to speak with other sensor gadgets like 3-pivot Magnetometer, pressure sensor, and so forth. It has an extra element of an on-chip temperature sensor.

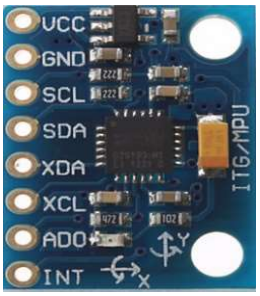


Fig. 14. MPU6050 MEMS Gyroscope [17]

2.3.4. BMP180 Pressure sensor

The BMP180 comprises a piezo-resistivity sensor (Fig. 15), a simple to advanced converter, and a control unit with E2PROM and a sequential I2C interface. The BMP180 conveys the uncompensated estimation of pressure and temperature. The microcontroller sends a beginning succession to begin a pressure or temperature estimation. In the wake of changing over time, the outcome esteem (weight

or temperature separately) can be perused through the I2C interface. For ascertaining temperature in °C and pressure in Pa, the adjustment information must be utilized. These constants can be perused out from the BMP180 E2PROM utilizing the I2C interface at programming initialization. The testing rate can be expanded up to 128 examples for every second (standard mode) for dynamic estimation. For this situation, it is adequate to quantify the temperature just once every second and utilize this incentive for all pressure estimations during a similar period [18].

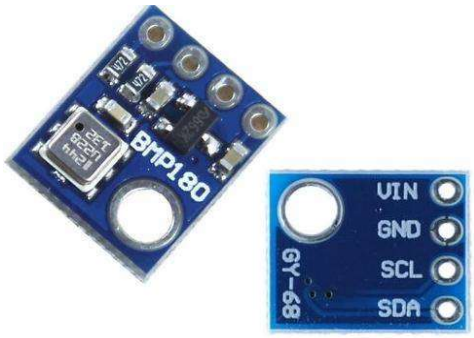


Fig. 15. BMP180 pressure sensor front and back view [19]

The BMP180 is small, cheap, and estimates both pressing factor and temperature since temperature changes the thickness of gasses like air. At higher temperatures, air isn't as thick and hefty, so it applies less tension on the sensor. At lower temperatures, air is thicker and gauges more, so it applies more tension on the sensor.

2.3.5. NEO 6MV2 GPS module

NEO – 6MV2 module is a group of independent GPS beneficiaries highlighting the superior u-blox 6 positioning engines. It is come with four connection Rx, Tx, Vcc and GND where Rx for receiver, Tx for transmission Vcc for power supply and GND for ground supply. These adaptable and practical collectors offer various network choices in a small 16 x 12.2 x 2.4 mm bundle. Their minimized engineering and force and memory alternatives make NEO-6 modules ideal for battery worked cell phones with exceptionally exacting expense and space requirements. In (Fig. 16) showing Neo 6mv2 gps module is showing.

Fig. 16. NEO 6MV2 GPS module [20]

NEO6MV2 has better power utilization. The Grove – GPS (Air530) has a super low power utilization at just 31uA, low force mode at 0.85 mA, which causes it to be the better GPS with lower power utilization.

3. Methods to connect sensors in microcontroller

There are several methods to connect sensors with raspberry pi 4B microcontroller, one of the methods explaining to connect all I2c sensors and GPS sensor with raspberry. To do so, you need to follow a few steps to enable the I2C protocol.

3.1. Steps to enable I2C protocol in raspberry pi 4B

- Step1 – log in to raspberry pi and open Raspberry command prompt.
- Step2 – Open the raspberry pi software configuration menu using command called (sudo raspi-config) and click enter.
- Step3 – In configuration window choose 3rd option that is called interfacing option and click enter (Fig. 17).
- Step4 – In interfacing option you will see another interface option then choose P5 I2C option (Fig. 18) and then enable it.
- Step5 – Use command (sudo reboot) and restart the system.

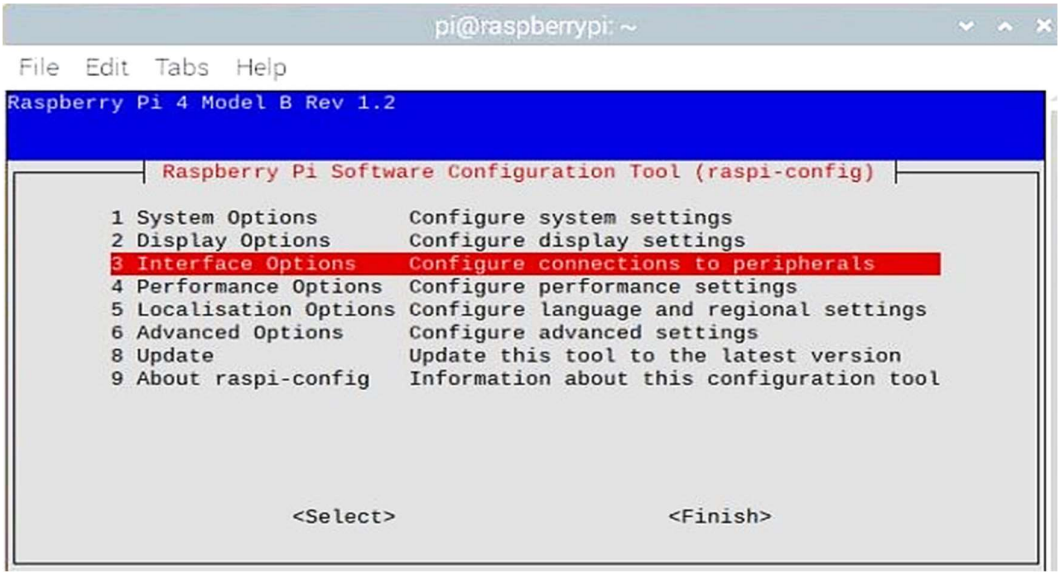


Fig. 17. Raspberry pi 4B configuration window

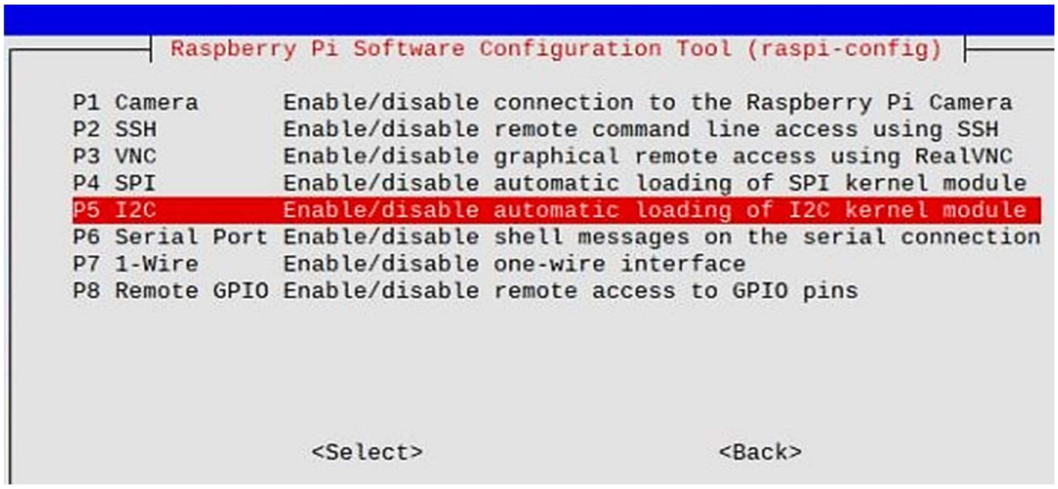


Fig. 18. Interfacing options window

3.2. Interference of MPU6050 gyroscope sensor with raspberry pi 4

Connection of MPU6050 with raspberry pi shown in (Fig. 19). In (Fig. 19), the red line represents power supply Vcc connected to 5-volt Vcc pin 4, the black line represents ground connection GND connected to GND pin 6, the blue line represents the serial clock connection CLS to raspberry pi Pin 3, and the yellow line represents the serial data connection SDA to SDA pin 5.

Once the connection of MPU6050 sensor is done with raspberry pi, install the library in the microcontroller. To install the library, open the command prompt in raspberry pi and write a command (sudo apt-get install python-smbus), and then use the command (pip install mpu6050-raspberry pi) to install the python library for MPU6050 sensor and install the library. After installing the library, check whether the connection of the sensor with the microcontroller is correct or not. To check the connection using the command (sudo i2cdetect -y 1) after this command, a list of devices connected to I2C will come up which will show the address of the sensor. If the address of the sensor showing that means, the connection is correct. In (Fig. 19) 68 is the address of the MPU6050 sensor.

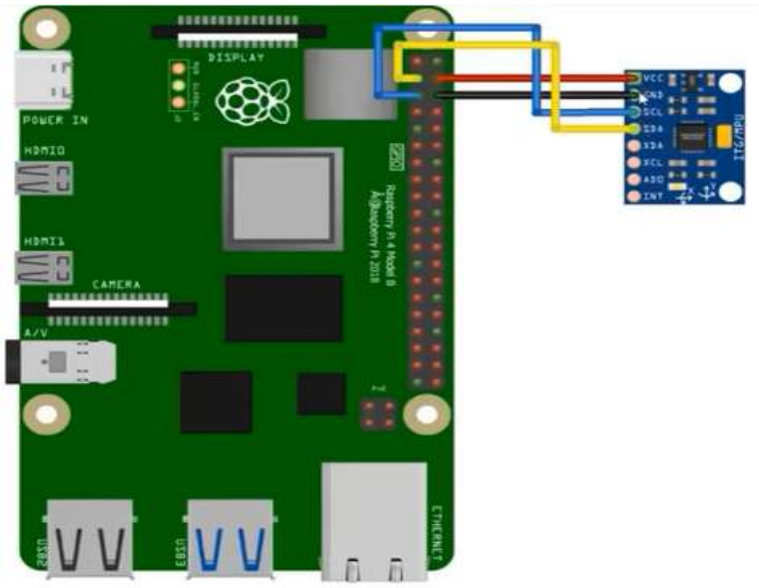


Fig. 19. Connection of MPU6050 gyroscope sensor with raspberry pi 4B

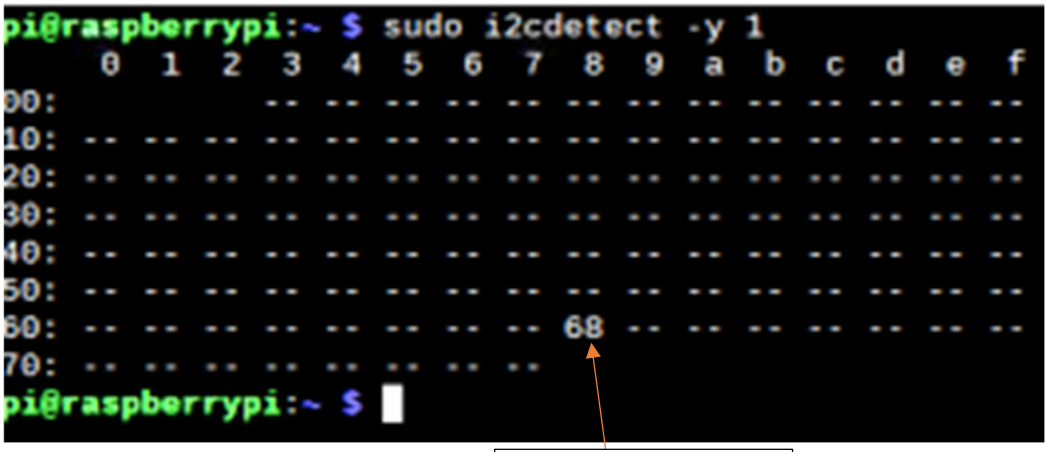


Fig. 20. MPU6050 sensor address in raspberry pi 4

This needs to be used using two MPU180 gyroscope sensors, and both sensors have the same configuration I2C device, so it will give the same address if connected parallelly through the breakout board or any other method. To overcome this problem, using a multiplexer, using multiplexer can create two different addresses of two same configure sensors, and using multiplexer could connect 8 I2C devices at the same time and can get data from all I2C devices. In (Fig. 21) shown the connection of two gyroscope sensors with raspberry pi through the multiplexer. Once all the connections are done, need to write program code to define each sensor address in the multiplexer. Code for multiplexer TCA9548A shown in (Fig. 22), while running the code can get all address of sensors connected in a microcontroller and can generate two different address of MPU6050. The address of the sensors shown in (Fig. 23), where 70 is the multiplexer address, 68 and 69 is the address of gyroscopes.

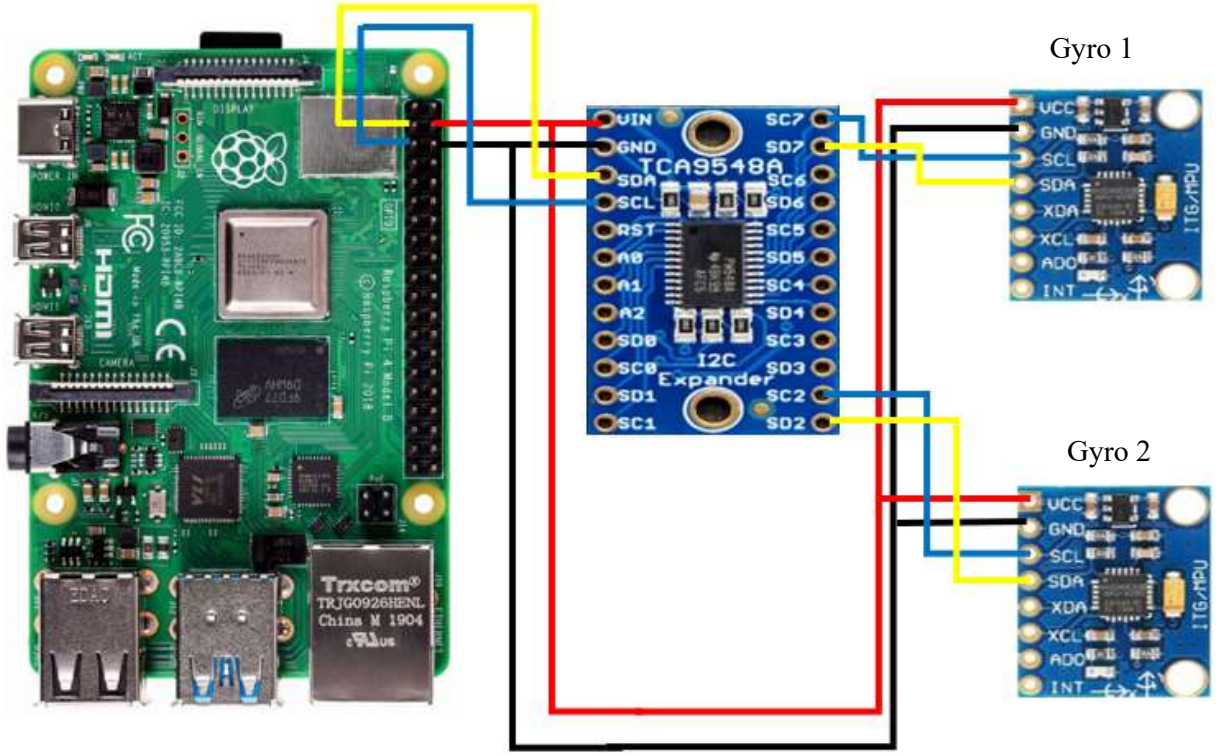


Fig. 21. Connection of two gyroscope sensors with raspberry pi 4 via TCA9548A multiplexer

Multiplexer TCA9548A is connected with raspberry pi 4 and two MPU6050 sensors (Fig. 21) where the red line represents Vcc connection, the black line represents ground connection, the yellow line represents SDA connection, and the blue line represents SCL connection. Gyro 1 SDA pin connected with multiplexer SD2 pin and SCL pin connected with SC2 pin whereas Gyro 2 SDA pin connected with multiplexer SD7 pin and SCL pin connected with SC7. There is no restriction to connect the SDA and SCL pin in any SD and SC pin of the multiplexer, but it should not connect with SD0 and SC0, the connection should be started from SC1 and SD1.

The python code for multiplexer used an algorithm to get the different addresses of the same configure of the I2C sensors. A device using two MPU6050 sensors and need to generate both sensor data requires two different address of MPU6050 to display both sensor data.

```
import smbus

class multiplex:

    def __init__(self, bus):
        self.bus = smbus.SMBus(bus)

    def channel(self, address=0x70,channel=0):  # values 0-3 indictae the channel, anything else (eg -1) turns off all channels

        if   (channel==0): action = 0x04
        elif (channel==1): action = 0x05
        elif (channel==2): action = 0x06
        elif (channel==3): action = 0x07
        elif (channel==4): action = 0x08
        else : action = 0x00

        self.bus.write_byte_data(address,0x04,action)  #0x04 is the register for switching channels

if __name__ == '__main__':

    bus=1       # 0 for rev1 boards etc.
    address=0x70

    plexer = multiplex(bus)
    plexer.channel(address,3)

    print "Now run i2cdetect"
```

Fig. 22. Python code for Multiplexer TCA9548A

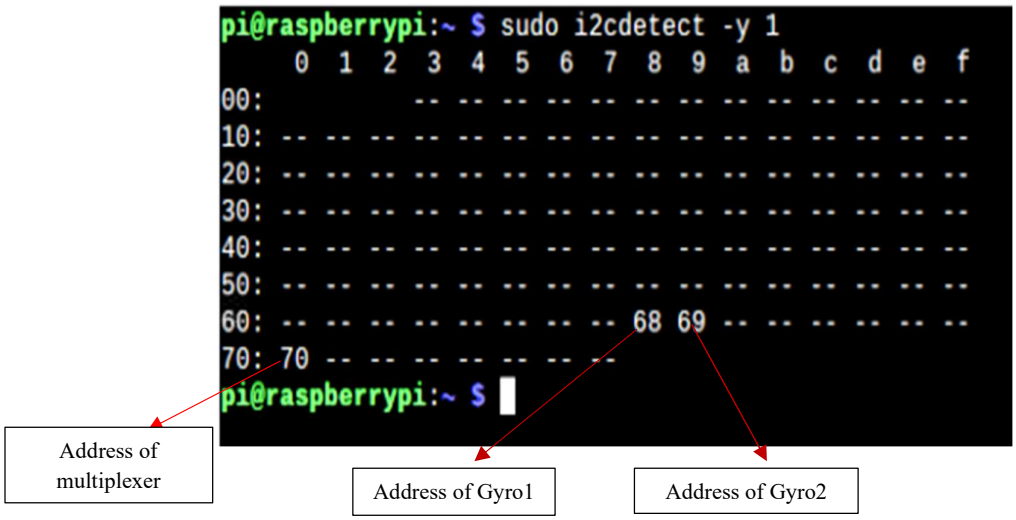


Fig. 23. Address of all sensors connected in microcontroller

3.3. Interference of NEO – 6 MV2 gps module sensor with raspberry pi 4B

Connection of NEO – 6MV2 shown in (Fig. 24), while connecting the GPs module be careful with few connections, here connect Rx input to raspberry pi Tx output (pin 8) represented by the yellow line and Tx input to raspberry pi Rx output (pin 10) represented by the yellow line, power supply represented by red line Vcc to 5 volt (pin 4) and the black line represents ground connection GND to GND (pin 6) as shown the connection of with raspberry pi 4B with NEO – 6MV2.

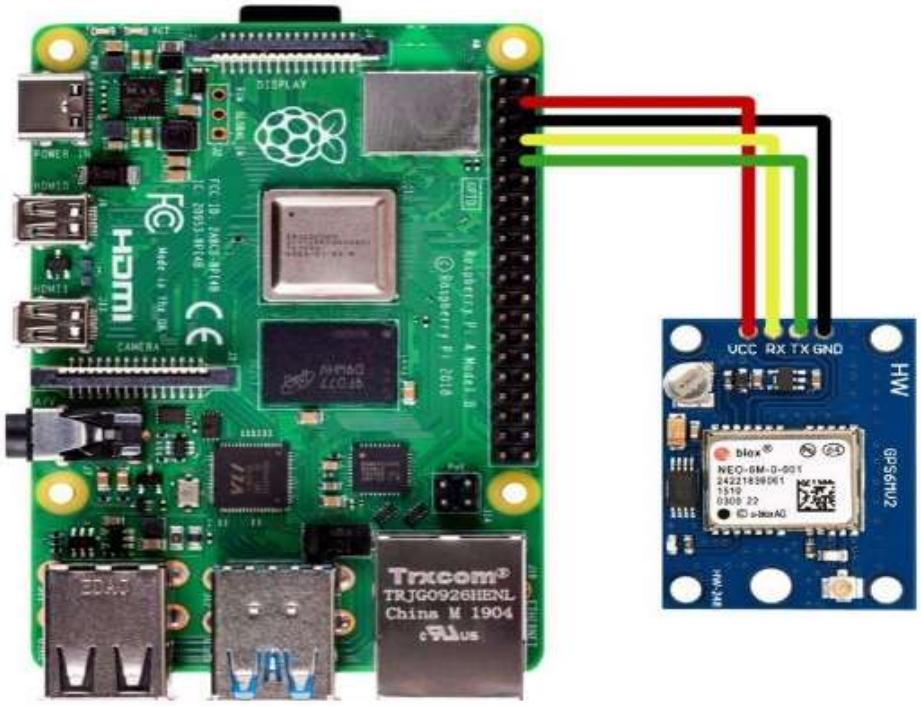


Fig. 24. Connection of NEO – 6MV2 gps module with raspberry pi 4B

When the connection is done, next need to configure raspberry pi's operating system to have the option to communicate with the GPS receiver. Start by running the command (sudo raspi-config) then in configuration window (Fig. 16) choose option 3 interface options then in next window showing in (Fig. 17) select P6 serial port and then select "Yes" in next pop-up dialog box "Would you like a login shell to be accessible over serial" and then next dialog box select "OK" and then back to main menu window of the raspi-config program, select finish and reboot raspberry pi.

```
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
$GPRMC,152702.00,A,2432.85106,N,07743.41624,E,0.660,,290421,,,A*70
$GPVTG,,T,,M,0.660,N,1.222,K,A*20
$GPGGA,152702.00,2432.85106,N,07743.41624,E,1,07,1.83,485.4,M,-52.5,M,,*7F
$GPGSA,A,3,29,15,18,13,24,20,23,,,,,,3.71,1.83,3.23*0E
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
```

Fig. 25. Output data from serial port

Write and run the python code shown in Appendix 1 to get the position, the library introduced in the last advance permits to communicate with the GPS daemon that then communicates with the GPS receiver. gpsd utilizes JSON objects to communicate with its client, so will get such JSON objects when utilizing the gpsd-client library, which can be parsed in our python script. While running the code will get the output like shown in (Fig. 25). It will give the longitude and latitude value. Using longitude and latitude, mobile applications will show the location and calculate the speed of walking or running. As the device is not moving anywhere, so the longitude and latitude not changing when the sensor moves from one place to another place, longitude and latitude will change when longitude and latitude change using those values can calculate speed.

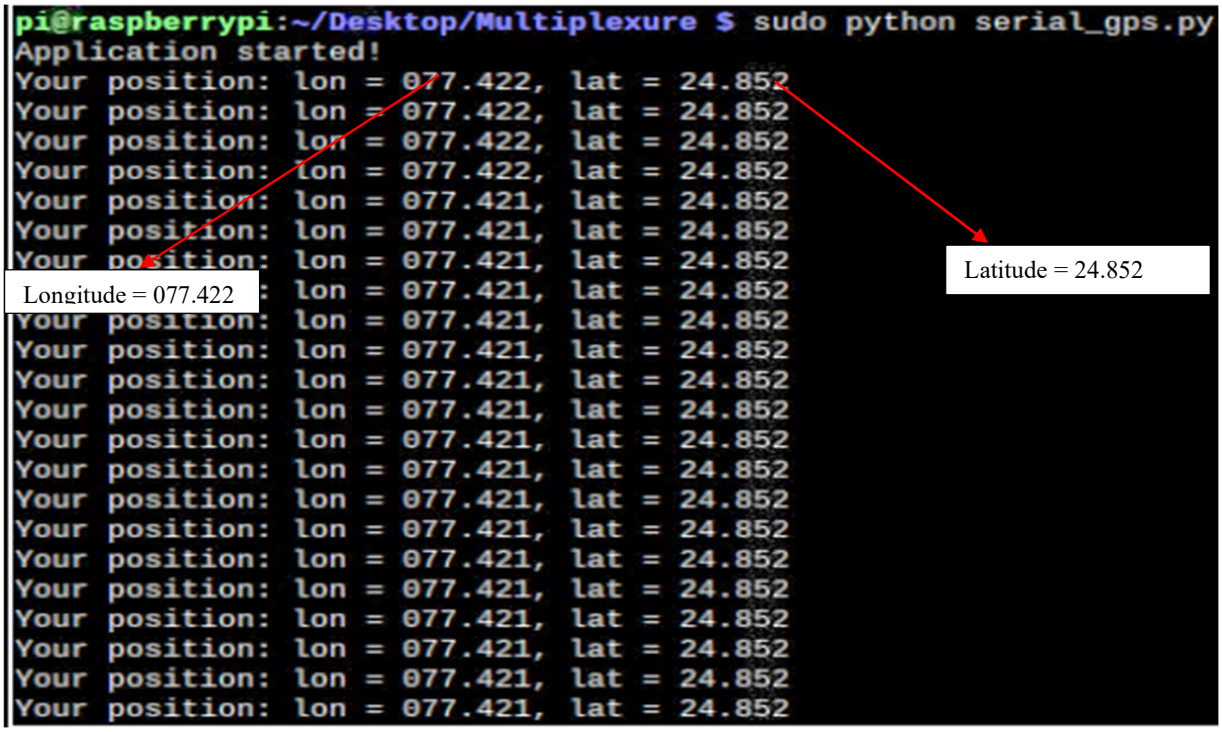


Fig. 26. Output data from gps module

Now install the gpsd and gpsd-client, use the command (sudo apt-get install gpsd gpsd-clients). Gpsd is an interfacing daemon for serial GPS recievers which upholds distinctive correspondence norms, it is utilized to get a test perusing and confirm that the equipment works accurately. When the establishment is done, check that you can get information from the GPS module. To do that, yield the information that it sends over the serial port, use a command (sudo cat /dev/serial0) then click enter, then the output will show something like shown in (Fig. 24). It does not matter what output you are getting if getting something like shown in (Fig. 24) means the GPS antenna is working, and it is able to receive signal and can generate GPS point longitude and latitude.

3.4. Interference of BMP180 presure sensor with raspberry pi 4

BMP180 sensor uses I2C protocol so connect SDA pin to raspberry pi SDA (pin 3), SCL pin to raspberry pi SCL (pin 5) and 3.3V (pin 1) power supply and GND to raspberry pi GND (pin 6) shown in (Fig. 26) and address of BMP180 sensor shown in (Fig. 27).

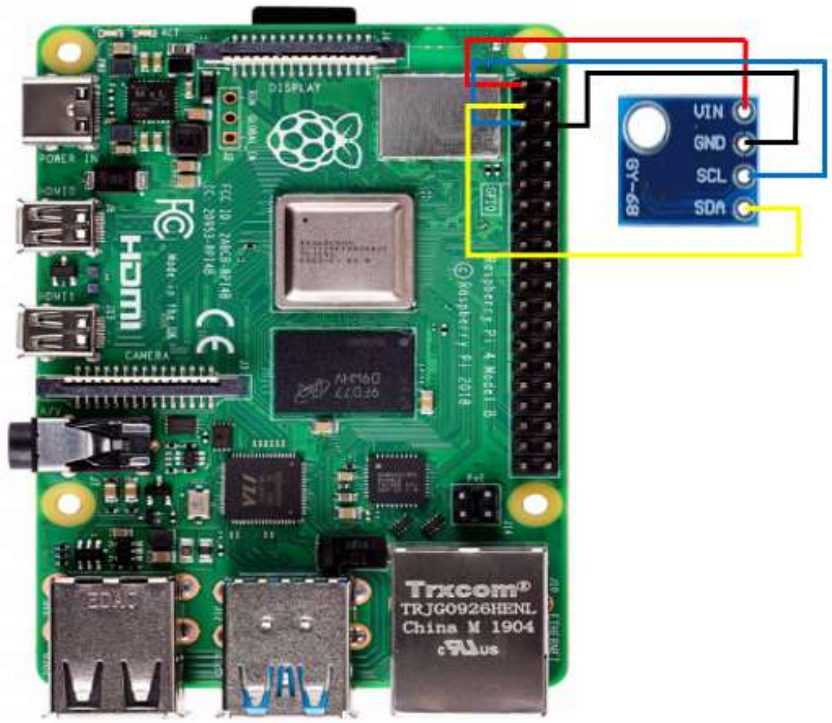


Fig. 27. Connection of BMP180 sensor with raspberry pi

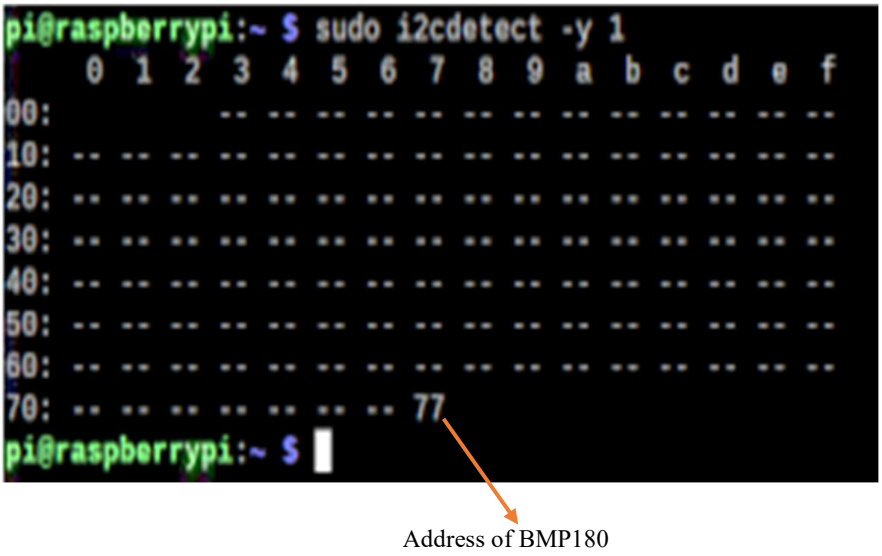


Fig. 28. Address of BMP180 pressure sensor

3.5. Connection of all the sensor and GPS module with raspberry Pi 4

Connection of all the sensors MPU6050, BMP180, and NEO-6MV2 GPS module shown in (Fig. 28) to get final output according to tasks, connection done with microcontroller pi through TCA9548A multiplexer. Multiplexer helps to generate different addresses of the same configuration of the I2C device. Using multiplexer, generating the different addresses of MPU6050. In (Fig. 29) showing the address of the sensors connected to the microcontroller, whereas the address of the NEO 6MV2 GPS module is not visible because raspberry uses UART as a serial console, and the GPSgps module connected in UART pin Tx and Rx, so it does not have an address like I2C devices.

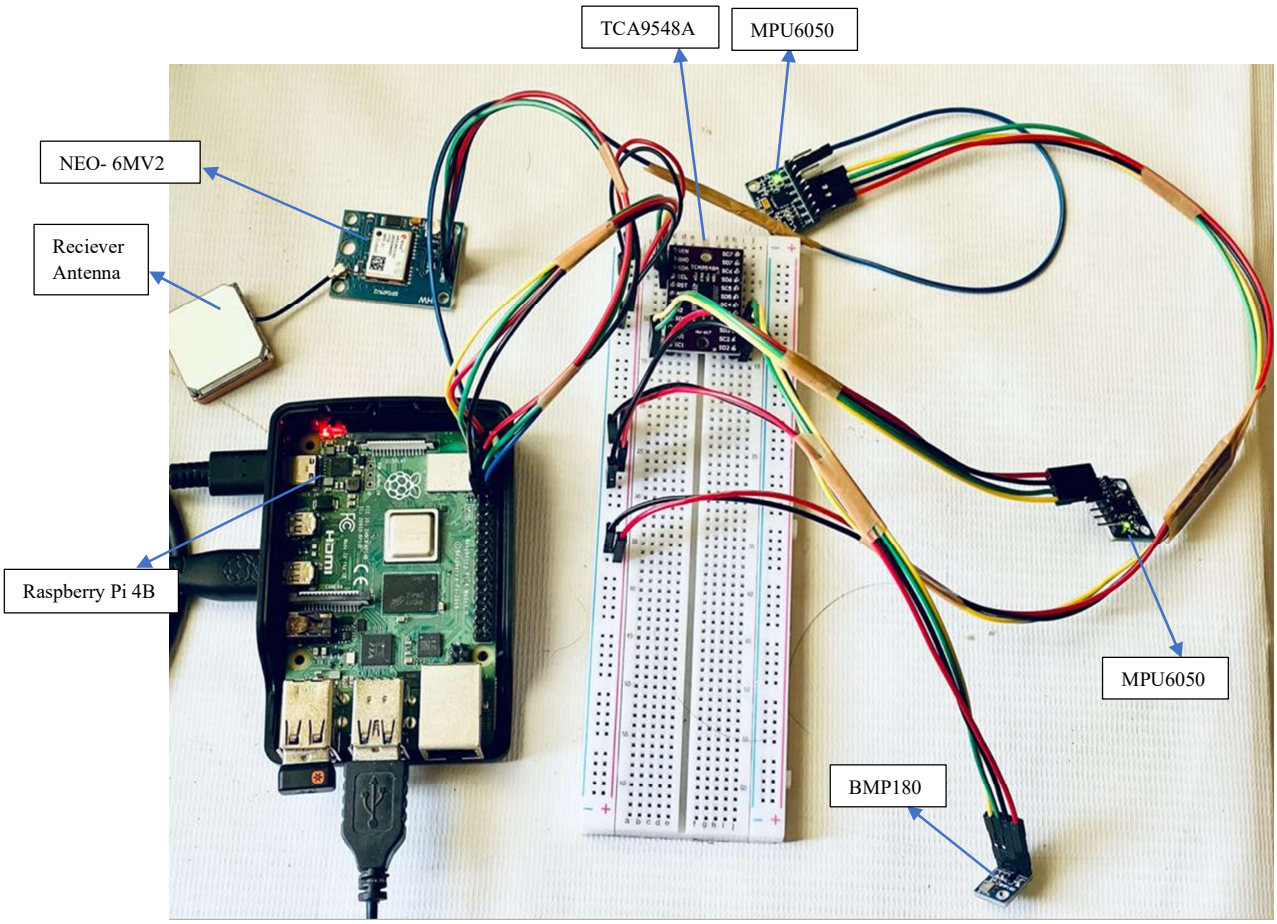


Fig. 29. Connection of sensors and GPS module for final device

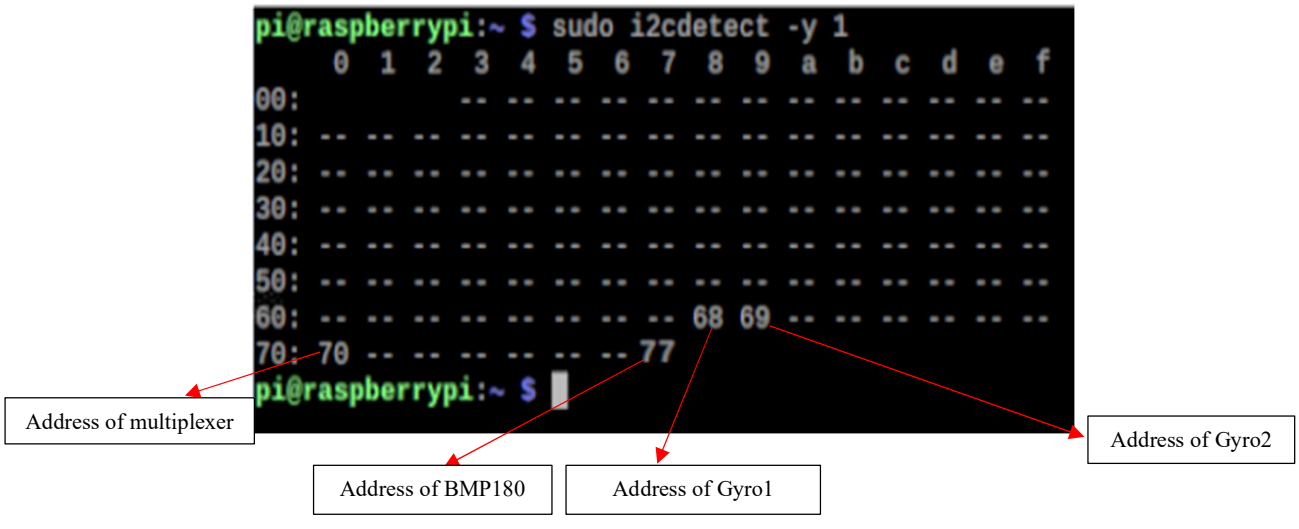


Fig. 30. Address of all the sensors connected in raspberry pi 4B

4. Methods to get output from the device

Once done with all the connections, then run the python code (Fig. 21) of the multiplexer to generate the address of I2C devices connected in a microcontroller, then write and run the python code available in Appendix 2 to get data from the device according to the task. In (Fig. 30) showing the output from the device. In the output, knee angle, step count, temperature, acceleration, and gyroscope data are available where Acc x1, Acc y1, Acc z1, and Gyro x1, Gyro y1, Gyro x1 are MPU6050 gyro 1 data whereas Acc x2, Acc y2, Acc z2, and Gyro x2, Gyro y2, Gyro x2 are MPU6050 gyro 2 data. MPU6050 sensors showing different values according to their position.

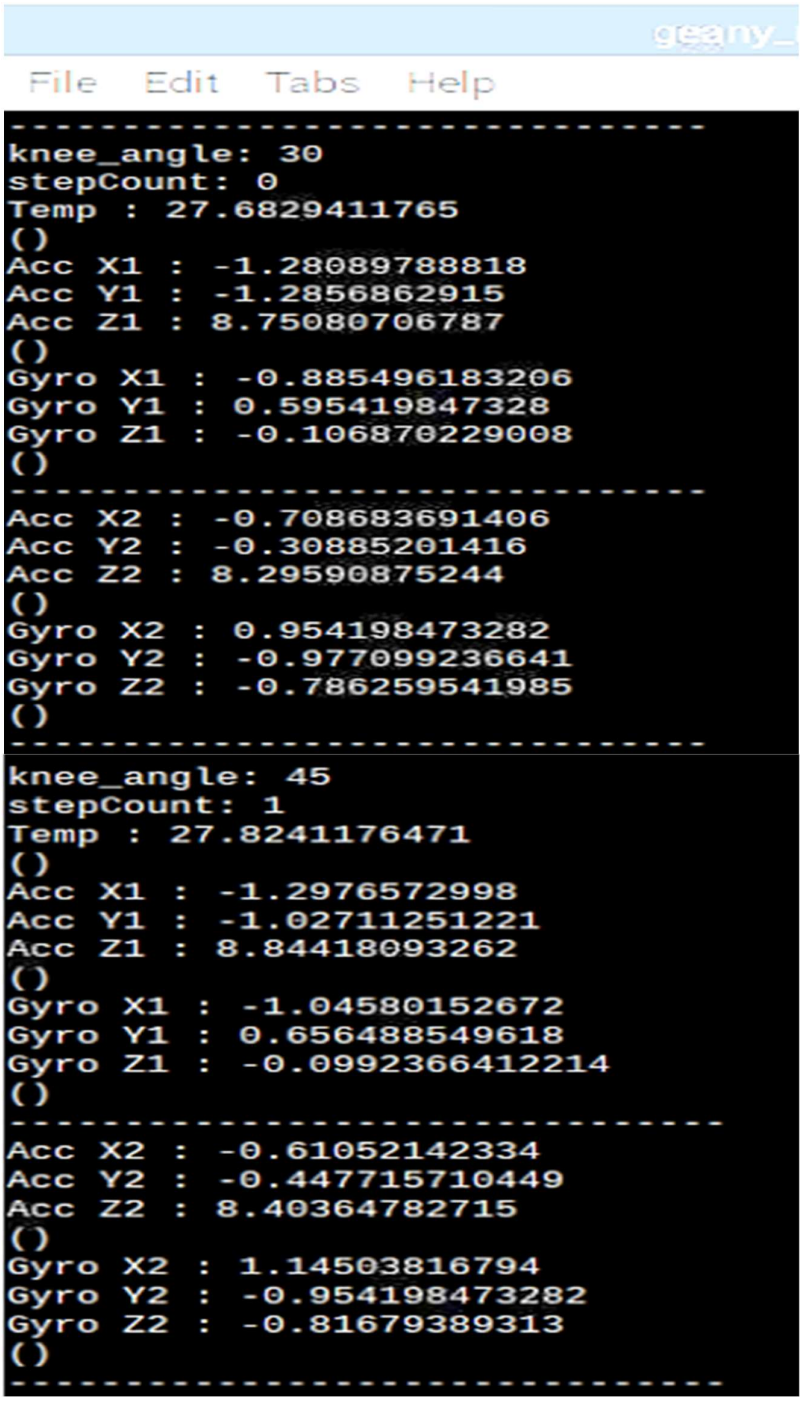


Fig. 31. Output from the setup device

Table 2. Output from the device

S.no	Knee Angle	Step Count	Temp	Acceleration1			Gyroscope1			Acceleration 2			Gyroscope 2		
				x1	y1	z1	x1	y1	z1	x2	y2	z2	x2	y2	z2
1	30	0	27.68	-1.28	-1.28	8.75	-0.88	0.59	-0.10	-0.70	-0.30	8.29	0.95	-0.97	-0.78
2	40	1	27.82	-1.29	-1.04	8.84	-1.04	0.65	-0.09	-0.61	-0.44	8.40	1.14	-0.97	-0.78
3	52	1	27.87	-1.30	0.79	8.69	-0.93	-0.34	-0.71	-0.71	-0.28	-8.42	1.35	-0.28	8.42
4	60	1	27.91	-1.27	-0.92	8.79	-0.88	0.91	-0.06	-0.58	-0.35	8.34	1.22	-0.70	-0.61
5	72	2	27.77	7.77	4.49	1.54	-34.22	-203.88	9.67	-3.56	-8.04	2.76	1.09	-1.06	-0.57
6	74	2	27.82	-3.73	2.54	19.61	15.90	209.26	2.75	-3.60	-8.10	2.79	1.77	-1.42	-0.59
7	87	2	27.73	-1.76	2.50	12.11	17.72	184.61	61.03	-3.62	-8.09	2.83	1.32	-0.93	-0.69
8	98	3	27.82	10.20	1.75	-1.23	89.28	156.45	62.58	-3.59	-8.05	2.72	1.18	-0.95	-0.58
9	99	3	27.87	-4.09	3.39	13.57	152.0	250.12	93.18	-3.58	-7.97	2.86	1.29	-0.92	-0.67
10	81	4	27.87	4.65	0.01	7.97	-0.97	0.11	-0.54	-2.46	-8.36	2.99	1.24	-1.02	-0.64
11	42	5	28.90	-5.26	0.12	7.04	-0.93	0.79	0.31	0.0	-7.62	4.90	1.0	-0.92	-0.74
12	59	5	28.15	-5.34	-0.07	6.97	-0.98	0.91	-0.04	0.026	-7.64	4.79	1.09	-0.93	-0.62
13	62	6	28.15	-5.30	-0.09	7.01	-0.96	0.79	-0.05	0.06	-7.61	4.93	1.29	-0.78	-0.63
14	85	7	28.20	-5.34	0.10	7.08	-0.94	0.82	-0.33	0.04	-7.64	4.98	1.09	-0.90	-0.87
15	76	7	29.8	3.36	-1.64	13.93	-46.5	-131.1	-8.95	-3.65	-8.07	2.77	1.36	-0.95	-0.74

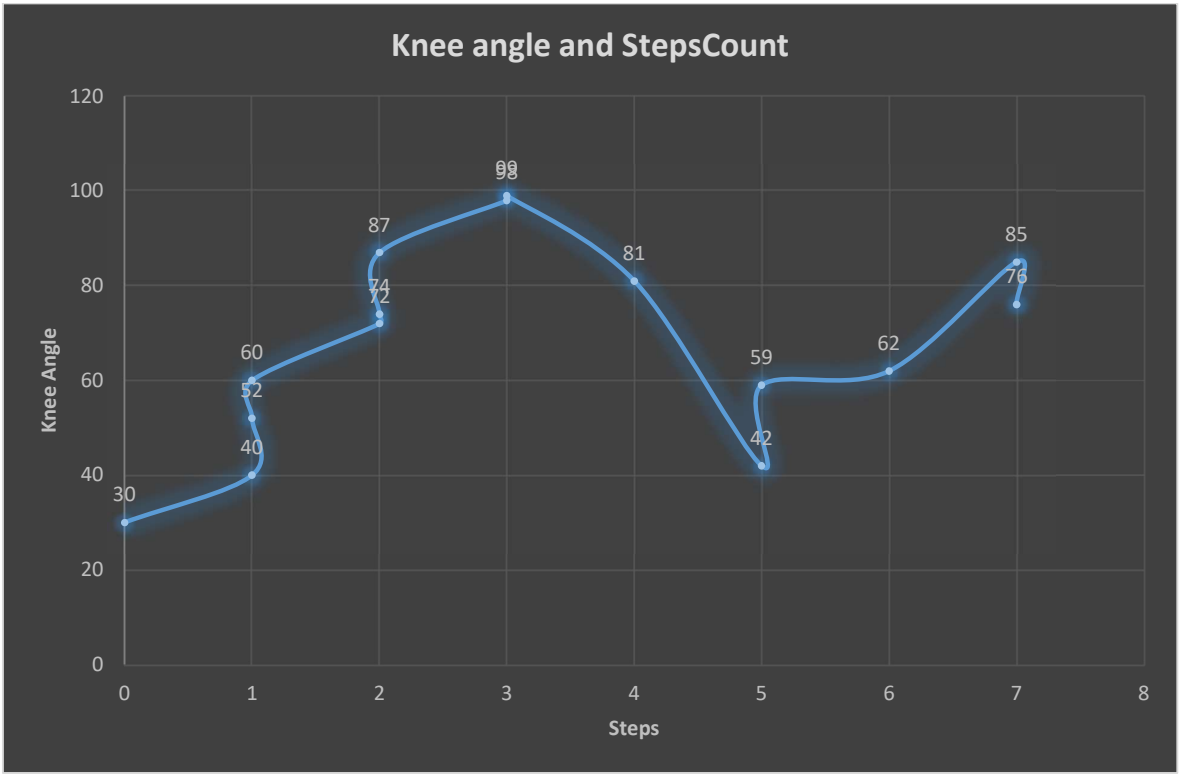


Fig. 32. Output data of smart knee brace

The output from the device shown in Table 2 using that data plotted the scatter graph to represent how many knees angles changed in between 0-10 walking steps. In Fig. 32, shows the graph of the output data where the x-axis is steps, and the y-axis is knee angle. All the value shown in Table 2 available in Appendix 5.

Knee angle calculation

In our set-up, the Pitch (У-pivot) point of the two sensors, which can be straightforwardly used to figure knee point, consistently falls in the ±90degree range. To guarantee the estimation is right paying little mind to a patient's position, the Roll (X-pivot) point is utilized to separate various positions. The Уaw (z-pivot) point isn't required for this calculation. Respects to the situating appeared in Equation (1) [21] knee point α is determined utilizing the accompanying recipe.

$$\alpha = \{180\text{-}Y_1+Y_2, \text{ if } |X_1| > 90 \text{ and } |X_2| > 90$$

$$360\text{-}Y_1\text{-}Y_2, \text{ if } |X_1| > 90 \text{ and } |X_2| \leq 90$$

$$Y_1+Y_2, \text{ if } |X_1| \leq 90 \text{ and } |X_2| > 90$$

$$180+Y_1\text{-}Y_2, \text{ if } |X_1| \leq 90 \text{ and } |X_2| \leq 90 \text{ ..... (1)}$$

Y_1, Y_2 and X_1 and X_2 are pitch and roll angles from each MPU6050 sensor (1 and 2) respectively.

Using the equation 1 wrote the code to calculate the knee angle. In (Fig. 33) shown the position of MPU6050 sensors to calculate the knee angle, α is the knee angle. In program for smart knee brace, algorithm (Fig. 34) used to calculate knee angle.

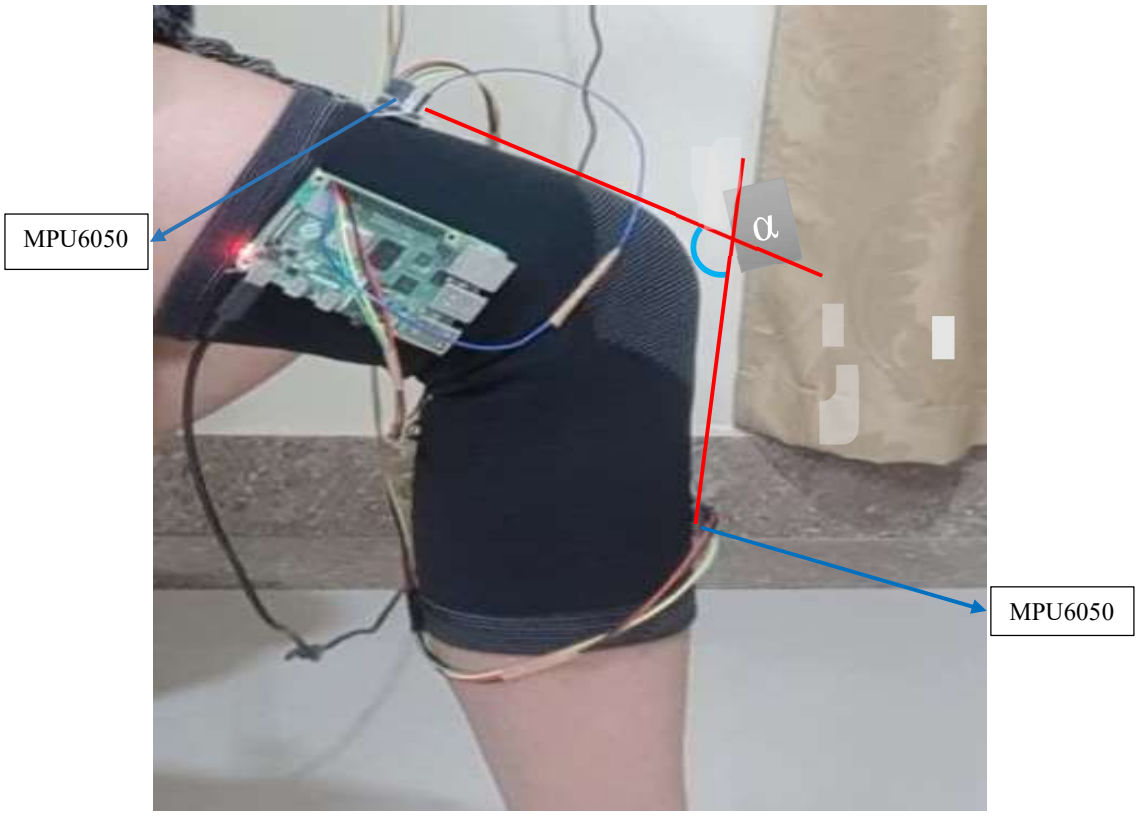


Fig. 33. Position of MPU6050 sensors to calculate knee angle

```
68      if(Total_angle_x>90 and Total_angle_x2>90 ):
69          knee_angle = 180 - Total_angle_y + Total_angle_y2
70      if(Total_angle_x>90 and Total_angle_x2<=90 ):
71          knee_angle = 360 - Total_angle_y - Total_angle_y2
72      if(Total_angle_x<=90 and Total_angle_x2>90 ):
73          knee_angle = Total_angle_y + Total_angle_y2
74      if(Total_angle_x<=90 and Total_angle_x2<=90):
75          knee_angle = 180 + Total_angle_y - Total_angle_y2
```

Fig. 34. Algorithm to calculate knee angle used in program

Speed Calculation

The equation for computing speed is speed equals distance covered divided by the time taken, regularly addressed as x = d/t.

Utilizing two GPS points (location) can compute the distance covered. GPS satellites send their location to collectors on the ground each second. By utilizing two GPS points can ascertain the distance covered. Utilizing the clock inside the GPS gadget (an extremely precise clock that synchronizes routinely with the nuclear timekeepers on board the GPS satellites) to quantify how long it required for the individual to go between those two points.

```python
def haversine_distance(lat1, lon1, lat2, lon2):
    r = 6371
    phi1 = np.radians(lat1)
    phi2 = np.radians(lat2)
    delta_phi = np.radians(lat2 - lat1)
    delta_lambda = np.radians(lon2 - lon1)
    a = np.sin(delta_phi / 2)**2 + np.cos(phi1) * np.cos(phi2) *   np.sin(delta_lambda / 2)**2
    res = r * (2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a)))
    return np.round(res, 2)

@app.route('/get_speed', methods=['GET', 'POST'])
def get_speed(time1, time2, lat1, lon1, lat2, lon2):
    distance_in_km = haversine_distance(lat1, lon1, lat2, lon2)
    timediff_hour = (time2 - time1).total_seconds() / 3660.0
    speed = distance_in_km/timediff_hour
    return speed

app.run(host="127.0.0.1", port="5000")
```

Fig 35. Algorithm to calculate speed

A GPS device in a smart knee brace records its location (scope and longitude, or lat/lon) at Point A and it observes the time too. A brief time later, it records its location once more (Point B). The GPS receiver would then be able to play out an estimation utilizing these numbers and decide the speed of the individual. In (Fig. 35) showing the algorithm to calculate speed which is written inside the API code.

5. Mobile application for device

The mobile application plays a key role for every smart device, it gives easy data access to users. To design the mobile application for a smart knee brace used React Native framework. Once done with the connection and getting the output data from the device, then need to store the data in the cloud, to store the data in the cloud using Amazon Web Services (AWS) wrote API code, and designed the mobile application to show the real-time data getting from the device. The process of a mobile application to display device data on a smartphone application is shown in the flow chart in (Fig. 36).

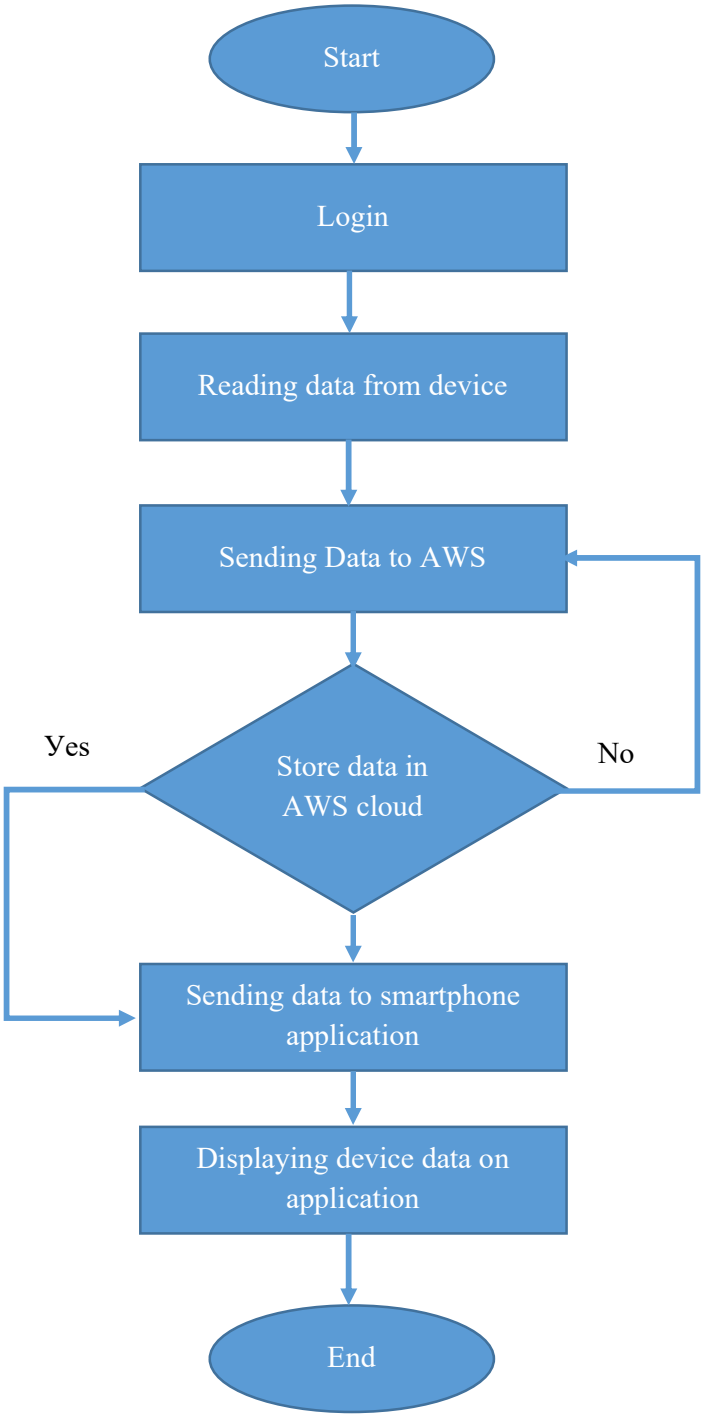


Fig. 36. Process of mobile application to display data

Table 3 showing data coming from the device. All the data store from the device and sending to the cloud using an application programming interface (API) and that data is shown in the mobile application.

Table 3. Device data showing in mobile application.

Data	**Unit**	**Data storage validity**
Knee Angle	Degree	12 hours
Walking steps	Number	12 hours
Speed	km/h	12 hours
Pressure	psi	12 hours
Temperature	Degree Celsius ℃	12 hours
Location of device	Longitude & Latitude	12 hours

Application Programming interface (API) code available in Appendix 3. API is a product delegate that permits two applications to converse with one another. An API is a lot of limits that grant applications to get to data and partner with external programming parts, working systems, or microservices, this API code will help to communicate knee brace with mobile application, this API will collect data from davice store in cloud and then share with mobile application. To work with APIs in Python, need apparatuses that will make those solicitations. In Python, the most well-known library for making solicitations and working with APIs is the requests library. The requests library isn't essential for the standard Python library. In API code Method GetSetData (GET API) and Method GetSetData (POST API) methods are used.

Method GetSetData (GET API): Every User has a unique user Id in backend, while calling GET API, we are passing User id in parameter, and filtering ortoticdevice table. Filters are based on User Id and Date-Time, we are getting records for the last 12 hours (All date-time has been saved in UTC Time)

Method GetSetData (POST API): Every User has a unique user Id in the backend while calling POST API, we are passing User id in the parameter if the user already exists, if the user does not exist then we are creating a unique user Id and assigning it to the user, and records have been inserted in orthoticdevice table with unique user Id and data time stamp (All date-time has been saved in UTC Time)

Table 4. Pages in mobile application and intelink pages with main page

Page	**Interlink page**	**Uses**
Login page	Sign up and Forgot pasword	Login page is available to log into application
Home Page	Edit Profile, Setting, dashboard and Logout	Home page is avaible to edit their profile, set limits for knee angle, walking speed and pressure
Dashboard page	Google map, Knee angle, Walking Speed and Pressure	Dashboard page is to see all data coming device in short view.

Ttable 4 showing the pages available in mobile application and which page interconnected with which pages, total 3 number of main pages are available and 7 number of interlink pages are available in mobile application.

5.1. Login and sign-up page of mobile application

In this section, users need to sign up and login into the application. In (Fig. 37) shown the login page that is the first page of the application where a new user can create a new account or log in using username or email and password and if the user forgot created password in the future then click on forgot password and user will get a link in the email to create a new password. To sign up, the user needs to fill in the personal information name, contact, email address, create a username, create a password, re-enter the password and device ID. When a user successfully created the account then they can log into the application using a username and password.

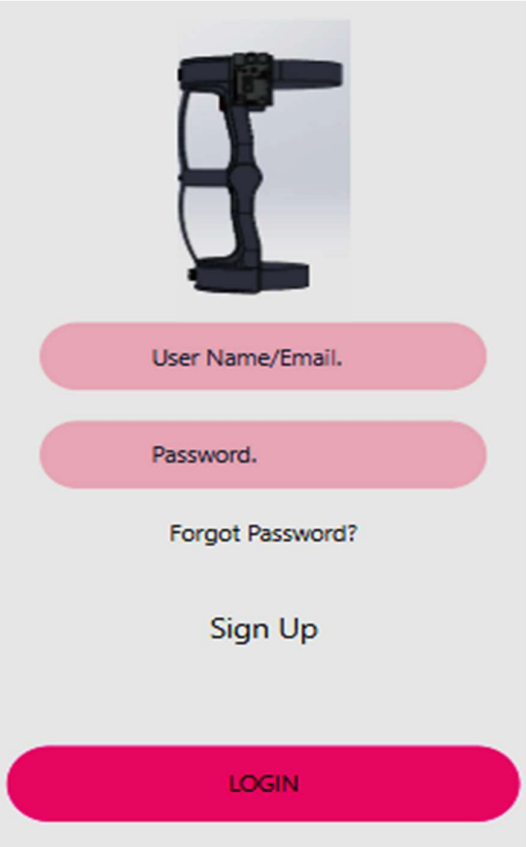


Fig. 37. Login page of mobile application

Sign up page shown in (Fig. 38) where all required field must fill by the user, when the required field filled by the user, then using the username and password user can log in then the user can see all the data from a device such as a knee angle, steps count, speed, pressure, and temperature. If user forgot their password, they could create a new password by click in on “Forgot password?” then a link will send to the user to register email address to create a new password. To create a new password user, need to write their registered email address on forgot password page as shown (Fig. 39) and in the email users also could see their username if they forgot. The link will share in email, by click in on the link user can create a new password.

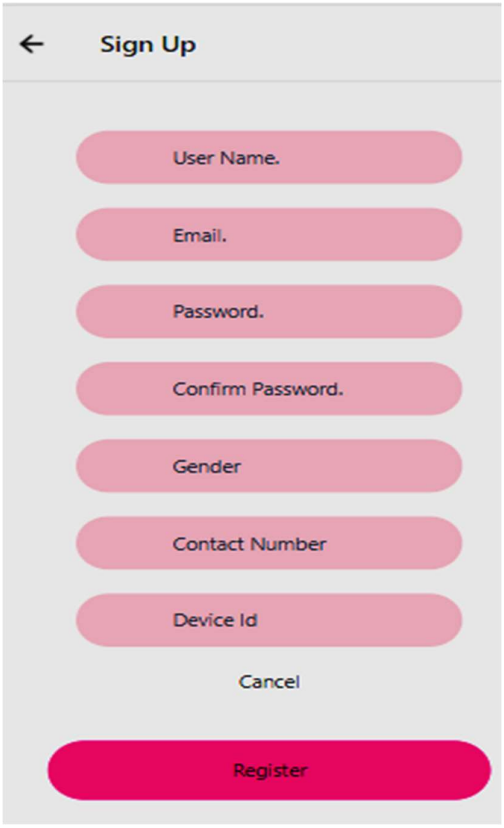


Fig. 38. Sign up page of mobile application

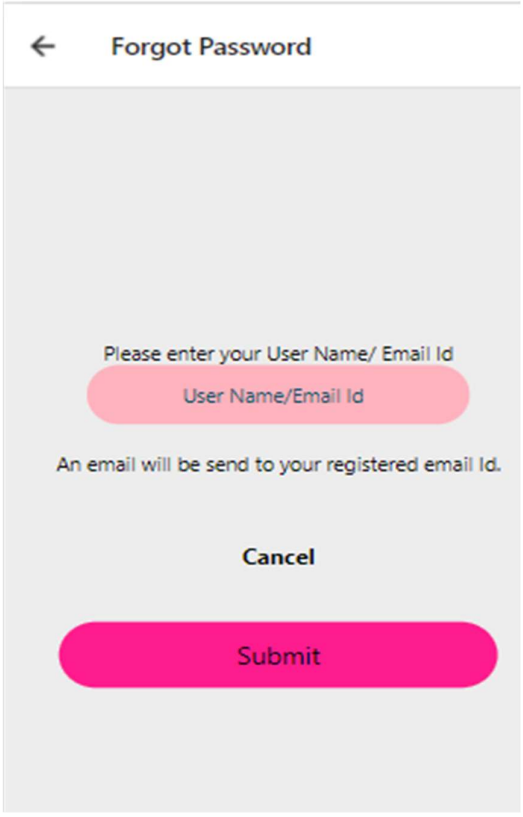


Fig. 39. Forgot password page

5.2. Welcome page of application

The welcome page is the home of mobile applications where user can edit their profile, set limits for knee angle, speed, and pressure, and can log out or can go to the dashboard to see knee brace data. In (Fig. 40) showing welcome page or home page.

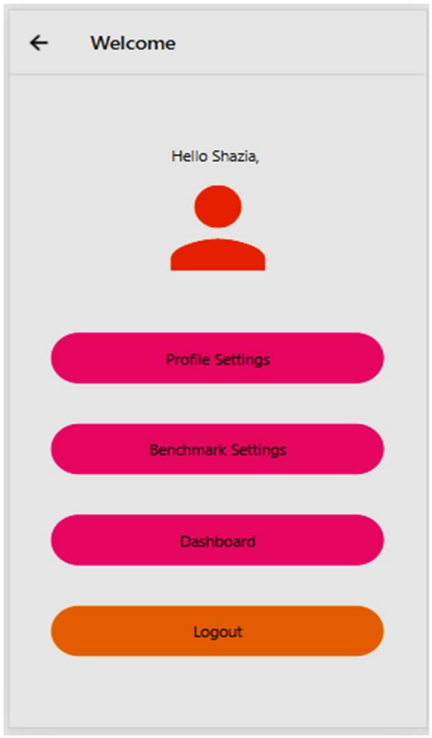


Fig. 40. Welcome page of mobile application

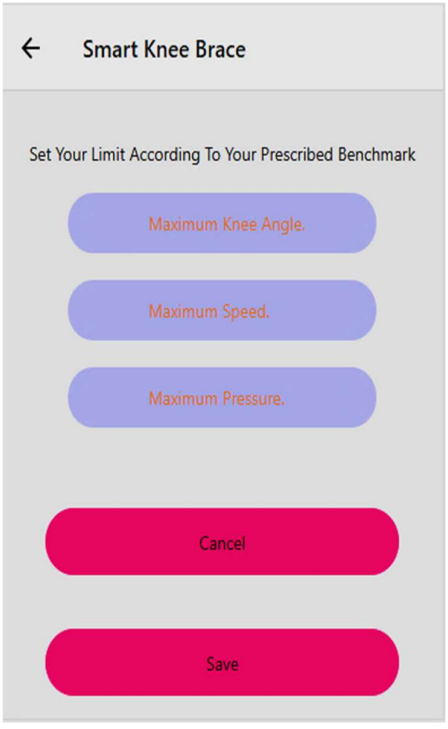


Fig 41. Benchmark setting page

Benchmark setting page showing (Fig. 41) where user can set their limit to get notification when they cross their limit. Profile setting page shown in (Fig. 42) where you can edit your profile.

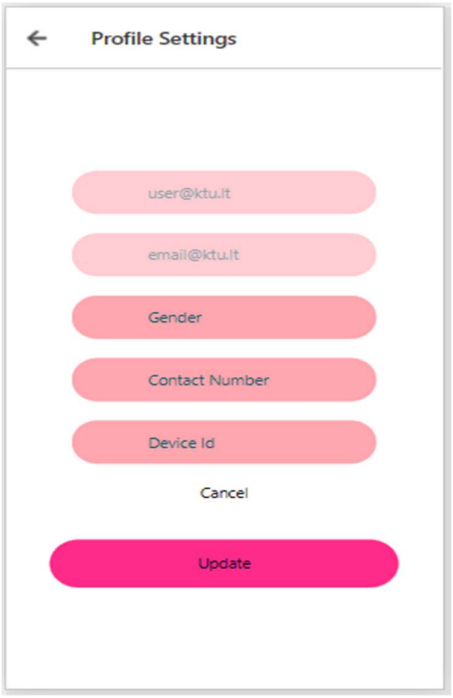


Fig. 42. Profile setting page

5.3. Dashboard of application

On the dashboard page, the user can see all the data coming from the smart knee brace by clicking on the name of the data. When the user clicks on the data name, the application will take in the main page of data where the user can see data coming from the device.

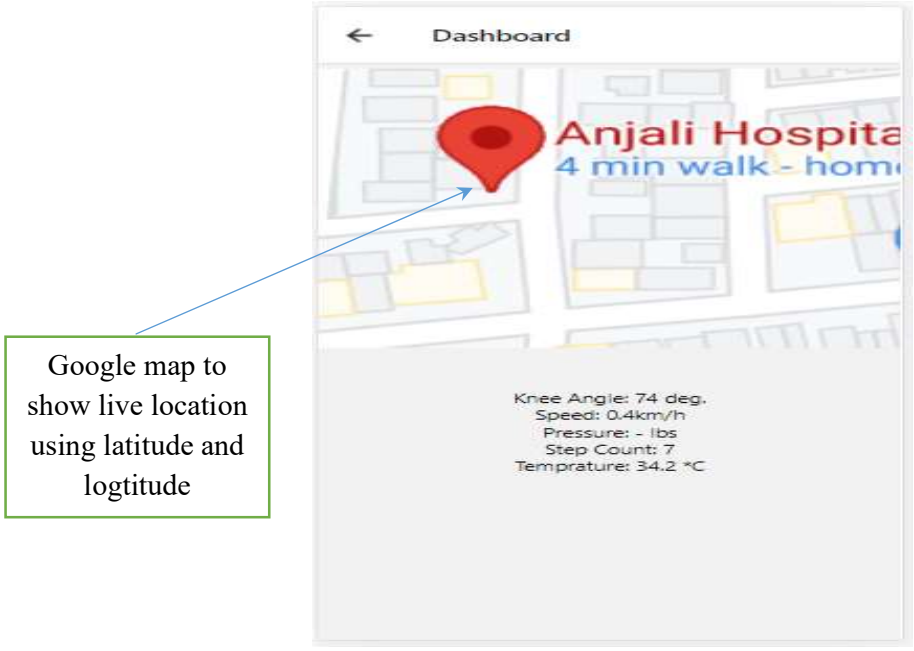


Fig. 43. Dashboard page of application

The dashboard page is shown in (Fig. 43) wherein the top of the page google map is available to show the live location of the knee brace using longitude and latitude and below the map knee angle, speed, pressure, steps count, and temperature are showing, when clicking on data, each data will take you in a new page and that page will show the graph-related your selected data stores in the duration of 12hours.

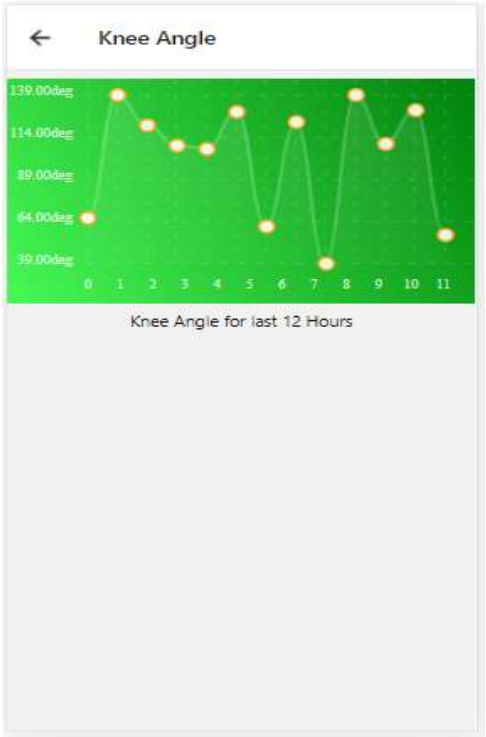


Fig. 44. Knee angle page of mobile application

Fig. 45. Walking speed page

Knee angle page is shown (Fig. 44) where the user can see the graph, that graph represents the changes in knee angle within 12 hours. In graph left side shows knee angle and the bottom line shows hours. In graph data showing linearly, knee angle should be changing every second but due to high data changes in every second and low storage capacity in cloud AWS, because of using free service of AWS, it is difficult to store big data and show knee angle changes in each second or minutes, so because low storage capacity, AWS storing hourly data according to written code in API and showing hourly data in the graph of knee angle and that data will be there for 12 hours, after 12 hours new graph will show. In the future, it is possible to show knee angle changes every second or minute for 12 hours or 24 hours if using paid service of AWS. The walking speed page is shown in (Fig. 45), where user can see their speed of walking in km/h in a total duration of 12 hours. On walking, the speed page left side showing speed in km/h and a bottom line representing the hours.

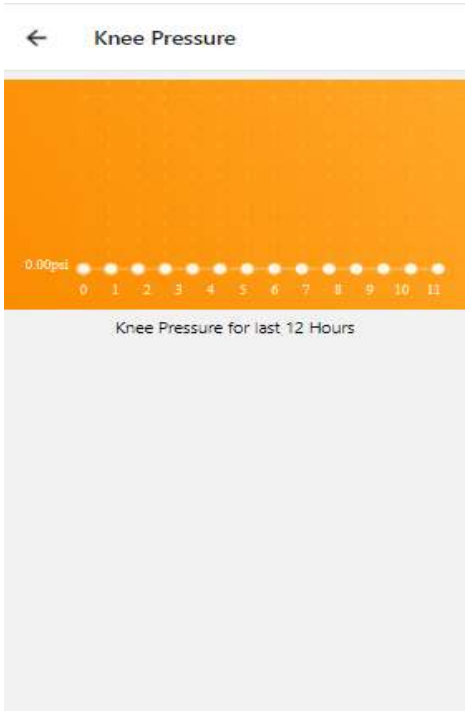


Fig. 46. Pressure page of mobile application

In (Fig. 46) shows pressure page which shows the variation of pressure in psi (pound per square). Due to a problem with the pressure sensor, the device was not able to generate pressure data, so the pressure page of the application is blank.

5.4. Program code of mobile application

Mobile application created in React-Native framework, it is a stage to make Android and IOS applications. React Native consolidates the most awesome aspects of local advancement with React, a top tier JavaScript library for building UIs. The program code of mobile application wrote in JavaScript language available in Appendix 4 can collect data from the cloud and shown in it the mobile application. React natives render to local stage user interface, which means the application utilizes a similar local stage APIs other application do. Respond makes stage explicit variants of segments so a solitary code base can share code across stages. With React Native, one group can uphold two stages and offer a typical innovation React.

Conclusions

1. Design of the device is based on the present design of hinged knee braces, which are generally used in knee osteoarthritis, and other knee problems and MPU6050 sensor, BMP180 sensor and NEOMV2 GPS module added on frame to make knee brace more advance.
2. The connection of MPU6050 gyroscope sensor, BMP180 pressure sensor, and NEO6MV2 GPS module was done and showing the address of the sensors in microcontroller and python code for smart knee brace showing temperature, gyroscope1, and gyrocope2 data, and using both gyroscope data measuring knee angle and counting walking steps.
3. API code is storing data in cloud and sending real time data to the mobile application and calculating speed using longitude and latitude value and sending speed data to mobile application.
4. Mobile application developed by using React native framework and java script, hence application was able to collect data from smart knee brace and displaying real time data of smart knee brace. Application tracking progress and screening user objectives during each activity session, each activity data store in mobile application for 12 hours after 12 hours it shows new data, in the dashboard of application google map is available which shows the live location of brace, application does not show graph for steps count and temperature it only shows the present data of steps count and temperature.

List of references

1. HUNT, M., BIRMINGHAM, T. and BRYANT, D., et al. *Types of knee braces and effects*. IEEE Consum, 2008.
2. DUIVENVOORDEN, T., et al. *Braces and Orthoses for Treating Osteoarthritis of the Knee.* John Wiley and Sons Ltd, March 2015.
3. REEVES, N.D. and BOWLING, F.L. Conservative Biomechanical Strategies for Knee Osteoarthritis. *Nature Reviews Rheumatology*, February 2011, vol. 7, no. 2. pp. 113-122 ISSN 1759-4790.
4. DENNIS, D.A., KOMISTEK, R.D., NADAUD, M.C. and MAHFOUZ, M. Evaluation of Off-Loading Braces for Treatment of Unicompartmental Knee Arthrosis. *Journal of Arthroplasty*, June 2006, vol. 21, no. 4 SUPPL. pp. 2-8 ISSN 0883-5403.
5. *How to Choose the Right Knee Brace | McDavid.* Available from: https://www.mcdavidusa.com/blogs/posts/how-to-choose-the-right-knee-brace.
6. *Benefits and Disadvantages of Knee Sleeves | Rxd Sleeves.* Available from: https://www.rxdsleeves.com/knee-sleeve-benefits/
7. SHULTZ, S.T., et al. *Knee Orthosis Orthoses for Knee Dysfunction*. Orthotics and Prosthetics in Rehabilitation (Fourth Edition), 2020.
8. OCHI, A., et al. *Custom-made Hinged Knee Braces with Extension Support can Improve Dynamic Balance.* Journal of Exercise Science and Fitness, December, 2018, vol. 16, no. 3. pp. 94-98.
9. GARGE, G. K., BALAKRISHNA, C. and DATTA, S. K. *Consumer health care: Current trends in consumer health monitoring.* IEEE Consum. Electron. Mag., 382 vol. 7, no. 1, pp. 38–46, January 2018.
10. *What is the Difference between an Open Patella and Closed Patella Knee – LP Supports.* Available from: https://www.lp-supports.com/blogs/injury-blog/what-is-the-difference-between-an-open-patella-and-closed-patella-knee-support.
11. XIONG, GUO. L., KAY, SOON, L. and TAHER, T. *Unrestrained Measurement of Arm Motion Based on a Wearable Wireless Sensor Network.* In Instrumentation and Measurement. IEEE Transactions on, 2010.
12. *X4 Smart Brace with Motion Intelligence _ DJO Global.* Available from: https://www.djoglobal.com/our-brands/donjoy/x4.
13. *Smart, Customizable Knee Orthosis - Today's Medical Developments.* Available from: https://www.todaysmedicaldevelopments.com/article/knee-orthosis-additive-manufacturing-smart-customizable-eos.
14. *Benchmarking the Raspberry Pi 4. Last Year's Release of the Raspberry Pi… | by Gareth Halfacree | Medium.* Available from: https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-4-73e5afbcd54b.
15. *Raspberry Pi GPIO Access | Raspberry Pi.* Available from: https://www.electronicwings.com/raspberry-pi/raspberry-pi-gpio-access.
16. *Amazon.Com: Pisugar2 Pro Portable 5000 mAh UPS Lithium Battery Power Module Platform for Every Raspberry Pi 3B/3B+/4B Model Accessories Handhold (Not Include Raspberry Pi): Computers & Accessories.* Available from:https://www.amazon.com/Pisugar2-Portable-Platform-Raspberry-Accessories/dp/B08D8PPCKN.

17. *MPU6050 Gyroscope Sensor | ThinkRobotics.In.* Available from:https://thinkrobotics.in/products/mpu6050-gyroscope-sensor?variant=16158094786632&currency=INR&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&gclid=Cj0KCQjw7pKFBhDUARIsAFUoMDZJE0-Q72yhq7SyUlmxCTEdo3wqjVpruCJz12KA2gUoQTTiPgJS-_oaAnv_EALw_wcB.
18. *Pressure Sensors for IoT - Infineon Technologies.* Available from: https://www.infineon.com/cms/en/product/sensor/pressure-sensors/pressure-sensors-for-iot/?gclid=Cj0KCQjw7pKFBhDUARIsAFUoMDYB0Sd1tmp5TnbuPXIeK89bElynKPHjLx17SiijHrCgfyfQ0S-DOI0aAt_VEALw_wcB&gclsrc=aw.ds.
19. *ESP8266 BMP180 Pressure Sensor Interface | Circuits4you.Com.* Available from: https://circuits4you.com/2019/03/23/esp8266-bmp180-pressure-sensor-interface/.
20. *GY-NEO6MV2 Flight Control GPS Module.* Available from:https://www.cytron.io/c-wireless-devices/p-gy-neo6mv2-flight-control-gps-module.
21. YANG QIU., et al. *Fun-Knee™: A Novel Smart Knee Sleeve for Total-Knee-Replacement Rehabilitation with Gamification.* IEEE, Apr 2017 Available from:https://ieeexplore.ieee.org/document/7939284.
22. SUSUMU, OTA., et al. *Effects of A Custom-Made Hinged Knee Brace with Knee Flexion Support for Patients with Knee Osteoarthritis*. Annual International Conference of the IEEE, 2015.
23. RAJA, K. et al. *Efficacy of knee braces and foot orthoses in conservative management of knee osteoarthritis*. A systematic review Am J Phys Med Rehabil. 2011; 90:247–262.
24. BERENGUERES, JOSE., et al. *A Smart Pressure-Sensitive Insole that Reminds You to Walk Correctly*. Member, EMBS. 2020.
25. BAGHAEI ROODSARI, R., et al. The Effect of Orthotic Devices on Knee Adduction Moment, Pain and Function in Medial Compartment Knee Osteoarthritis: A Literature Review. *Disability and Rehabilitation: Assistive Technology*, Jul 04, 2017, vol. 12, no. 5. pp. 441-449.
26. AKRIBOPOULOS O., et al. *Building a Platform-Agnostic Wireless Network of Interconnected Smart Objects*. In Informatics (PCI), 15th Panhellenic Conference on, 2011
27. DEJNABADI, H. B.M. JOLLES, and K. AMINIAN., et al. *A new approach to accurate measurement of uniaxial joint angles based on a combination of accelerometers and gyroscopes*. Biomedical Engineering, IEEE Transactions on, 2005. 52(8): p. 1478-1484.
28. JIA, M., et al. *Design and Evaluation of Dynamic Knee Orthosis System for Females with Knee Ligament Injuries.* Cornell University Library, Jan 01, 2017.
29. YANG QIU, ENG CHUAN NEOH, HUIGUO ZHANG, XIN YUE KHAW, XIUYI FAN, and CHUNYAN MIAO. *A Novel Smart Knee Sleeve for Total-Knee-Replacement Rehabilitation with Gamification*. Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly, 2015.
30. LEWIS, H.E, et al. *Hinged knee brace having torque pads for producing inward support pressure.* March, 2018.

Appendices

Appendix 1. Python code to get GPS location

```
import serial

SERIAL_PORT = "/dev/serial0"
running = True

def formatDegreesMinutes(coordinates, digits):

    parts = coordinates.split(".")

    if (len(parts) != 2):
        return coordinates

    if (digits > 3 or digits < 2):
        return coordinates

    left = parts[0]
    right = parts[1]
    degrees = str(left[:digits])
    minutes = str(right[:3])

    return degrees + "." + minutes


def getPositionData(gps):
    data = gps.readline()
    message = data[0:6]
    if (message == "$GPRMC"):
        parts = data.split(",")
        if parts[2] == 'V':
            print "GPS receiver warning"
        else:
            longitude = formatDegreesMinutes(parts[5], 3)
            latitude = formatDegreesMinutes(parts[3], 2)
            print "Your position: lon = " + str(longitude) + ", lat = " + str(latitude)
    else:
        pass

print "Application started!"
gps = serial.Serial(SERIAL_PORT, baudrate = 9600, timeout = 0.5)

while running:
    try:
        getPositionData(gps)
    except KeyboardInterrupt:
        running = False
        gps.close()
        print "Application closed!"
    except:
        print "Application error!"

```

Appendix 2. Python code for smart knee brace

```python
from mpu6050 import mpu6050
import time
import math
import random

mpu1 = mpu6050(0x68)
mpu2 = mpu6050(0x69)
knee_angle = 0
elapsedTime = 10

MagnitudePrevious = 0
stepCount = 0

while True:
    print("Temp : "+str(mpu1.get_temp()))
    print()

    accel_data = mpu1.get_accel_data()
    print("Acc X1 : "+str(accel_data['x']))
    print("Acc Y1 : "+str(accel_data['y']))
    print("Acc Z1 : "+str(accel_data['z']))
    print()

    gyro_data = mpu1.get_gyro_data()
    print("Gyro X1 : "+str(gyro_data['x']))
    print("Gyro Y1 : "+str(gyro_data['y']))
    print("Gyro Z1 : "+str(gyro_data['z']))
    print()
    print("------------------------------")

    accel_data2 = mpu2.get_accel_data()
    print("Acc X2 : "+str(accel_data2['x']))
    print("Acc Y2 : "+str(accel_data2['y']))
    print("Acc Z2 : "+str(accel_data2['z']))
    print()

    gyro_data2 = mpu2.get_gyro_data()
    print("Gyro X2 : "+str(gyro_data2['x']))
    print("Gyro Y2 : "+str(gyro_data2['y']))
    print("Gyro Z2 : "+str(gyro_data2['z']))
    print()
    print("------------------------------")
    time.sleep(1)

    Gyro_angle_x = float(gyro_data['x'])*elapsedTime;
    Gyro_angle_y = float(gyro_data['y'])*elapsedTime;

    Gyro_angle_x2 = float(gyro_data2['x'])*elapsedTime;
    Gyro_angle_y2 = float(gyro_data2['y'])*elapsedTime;

    #print("Gyro_angle_x: " + str(Gyro_angle_x))
    #print("Gyro_angle_y: " + str(Gyro_angle_y))


    Acc_angle_x = (math.atan((float(accel_data['x']))/math.sqrt(pow((float(accel_data['x'])),2) + pow((float(accel_data['z'])),2)))*180);
    Acc_angle_y = (math.atan(-1*(float(accel_data['x']))/math.sqrt(pow((float(accel_data['y'])),2) + pow((float(accel_data['z'])),2)))*180);

    Acc_angle_x2 = (math.atan((float(accel_data2['x']))/math.sqrt(pow((float(accel_data2['x'])),2) + pow((float(accel_data2['z'])),2)))*180);
    Acc_angle_y2 = (math.atan(-1*(float(accel_data2['x']))/math.sqrt(pow((float(accel_data2['y'])),2) + pow((float(accel_data2['z'])),2)))*180);
```

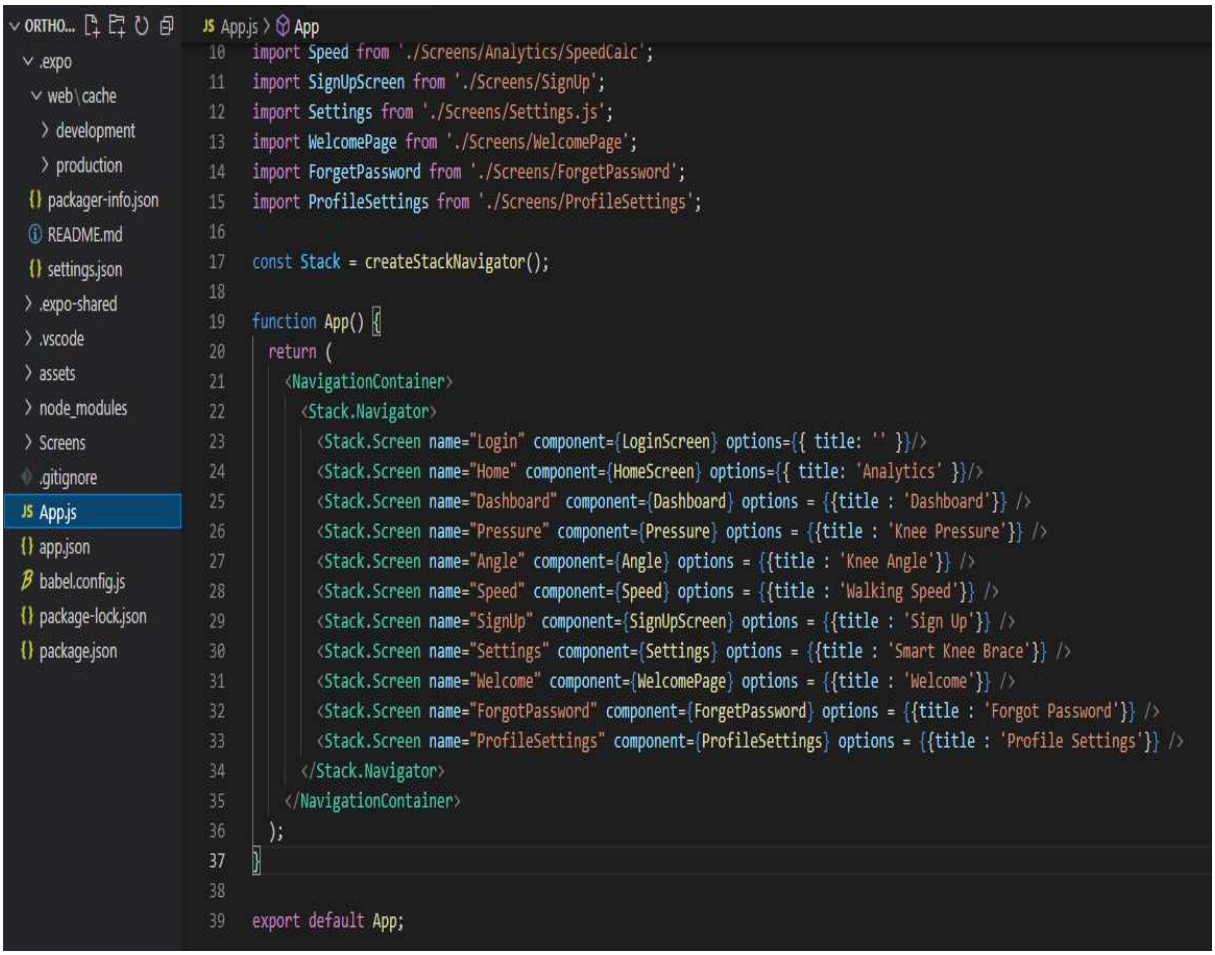
```
60
61      Total_angle_x = 0.98 *(0 + Gyro_angle_x) + 0.02*Acc_angle_x;
62      Total_angle_y = 0.98 *(0 + Gyro_angle_y) + 0.02*Acc_angle_y;
63
64      Total_angle_x2 = 0.98 *(0 + Gyro_angle_x2) + 0.02*Acc_angle_x2;
65      Total_angle_y2 = 0.98 *(0 + Gyro_angle_y2) + 0.02*Acc_angle_y2;
66
67
68      if(Total_angle_x>90 and Total_angle_x2>90 ):
69          knee_angle = 180 - Total_angle_y + Total_angle_y2
70      if(Total_angle_x>90 and Total_angle_x2<=90 ):
71          knee_angle = 360 - Total_angle_y - Total_angle_y2
72      if(Total_angle_x<=90 and Total_angle_x2>90 ):
73          knee_angle = Total_angle_y + Total_angle_y2
74      if(Total_angle_x<=90 and Total_angle_x2<=90):
75          knee_angle = 180 + Total_angle_y - Total_angle_y2
76
77      accel_data = mpu1.get_accel_data()
78      acc_x = float(accel_data['x'])
79      acc_y = float(accel_data['y'])
80      acc_z = float(accel_data['z'])
81
82      Magnitude = math.sqrt(acc_x*acc_x + acc_y*acc_y + acc_z*acc_z)
83      MagnitudeDelta = float(Magnitude)-float(MagnitudePrevious)
84      MagnitudePrevious = Magnitude
85      print("stepCount: " + str(stepCount))
86      if MagnitudeDelta > 6:
87          stepCount = stepCount + 1
88          pass
89
```

Appendix 3. API code for mobile application

C: > Users > Shazia Khan > Desktop > APIs > orthoticdevice_api.py

```
from flask import Flask,render_template, request
app = Flask(__name__)
from sqlalchemy import  create_engine

engine = create_engine("mysql+pymysql://admin:Focus1234@orthoticdevice.cbrsnf3doki2.us-east-2.rds.amazonaws.com/orthoticdevice")

@app.route('/ping', methods=['GET'])
def home():
    if (request.method == 'GET'):
        return 'PONG'

##This Mehod get and set orthotic device data based on user Id.

@app.route('/GetSetData', methods=['GET', 'POST'])
def GetSetData():
    if (request.method == 'GET'):
        datarequested = request.data
        query = (model.Session.query(orthoticdevicedata)
        .filter_by(userId=datarequested.userId)
        .filter_by(datetime > DATE_ADD(DateTime.Now(), INTERVAL -12 HOUR))
        .order_by(datetime.desc())
        )
        return json.dumps(query)
    else:
        if not request.json or not 'userId' in request.json:
        abort(400)
        db.session.add(datarequested)
        db.session.commit()
        return jsonify({'orthoticdevicedata': orthoticdevicedata.serialize()}), 201

def haversine_distance(lat1, lon1, lat2, lon2):
   r = 6371
   phi1 = np.radians(lat1)
   phi2 = np.radians(lat2)
   delta_phi = np.radians(lat2 – lat1)
   delta_lambda = np.radians(lon2 – lon1)
   a = np.sin(delta_phi / 2)**2 + np.cos(phi1) * np.cos(phi2) *   np.sin(delta_lambda / 2)**2
   res = r * (2 * np.arctan2(np.sqrt(a), np.sqrt(1 – a)))
   return np.round(res, 2)

@app.route('/get_speed', methods=['GET', 'POST'])
def get_speed(time1, time2, lat1, lon1, lat2, lon2):
    distance_in_km = haversine_distance(lat1, lon1, lat2, lon2)
    timediff_hour = (time2 - time1).total_seconds() / 3660.0
    speed = distance_in_km/timediff_hour
    return speed

app.run(host="127.0.0.1", port="5000")
```

Appendix 4. Program code for mobile application



Appendix 5. Output data of smart knee brace

geany_run_script_EITE30.sh

File Edit Tabs Help

```
--------------------------------
knee_angle: 30
stepCount: 0
Temp : 27.6829411765
()
Acc X1 : -1.28089788818
Acc Y1 : -1.2856862915
Acc Z1 : 8.75080706787
()
Gyro X1 : -0.885496183206
Gyro Y1 : 0.595419847328
Gyro Z1 : -0.106870229008
()
--------------------------------
Acc X2 : -0.708683691406
Acc Y2 : -0.30885201416
Acc Z2 : 8.29590875244
()
Gyro X2 : 0.954198473282
Gyro Y2 : -0.977099236641
Gyro Z2 : -0.786259541985
()
--------------------------------
knee_angle: 45
stepCount: 1
Temp : 27.8241176471
()
Acc X1 : -1.2976572998
Acc Y1 : -1.02711251221
Acc Z1 : 8.84418093262
()
Gyro X1 : -1.04580152672
Gyro Y1 : 0.656488549618
Gyro Z1 : -0.0992366412214
()
--------------------------------
Acc X2 : -0.61052142334
Acc Y2 : -0.447715710449
Acc Z2 : 8.40364782715
()
Gyro X2 : 1.14503816794
Gyro Y2 : -0.954198473282
Gyro Z2 : -0.81679389313
()
--------------------------------
knee_angle: 52
stepCount: 1
Temp : 27.8711764706
()
Acc X1 : -1.30962830811
Acc Y1 : -0.799663354492
Acc Z1 : 8.69574042969
()
Gyro X1 : -0.93893129771
Gyro Y1 : 0.656488549618
Gyro Z1 : -0.343511450382
()
--------------------------------
Acc X2 : -0.715866296387
Acc Y2 : -0.287304199219
Acc Z2 : 8.42519564209
()
Gyro X2 : 1.35877862595
Gyro Y2 : -0.893129770992
Gyro Z2 : -0.740458015267
()
--------------------------------
```

geany_run_script_Q09C30.sh

File Edit Tabs Help

```
------------------------------
knee_angle: 60
stepCount: 1
Temp : 27.9182352941
()
Acc X1 : -1.27850368652
Acc Y1 : -0.92655604248
Acc Z1 : 8.79869110107
()
Gyro X1 : -0.885496183206
Gyro Y1 : 0.916030534351
Gyro Z1 : -0.0610687022901
()
------------------------------
Acc X2 : -0.581791003418
Acc Y2 : -0.359130249023
Acc Z2 : 8.34379278564
()
Gyro X2 : 1.2213740458
Gyro Y2 : -0.702290076336
Gyro Z2 : -0.610687022901
()
------------------------------
knee_angle: 72
stepCount: 2
Temp : 27.7770588235
()
Acc X1 : 7.77636699219
Acc Y1 : 4.49870491943
Acc Z1 : 1.54904847412
()
Gyro X1 : -34.2213740458
Gyro Y1 : -203.381679389
Gyro Z1 : 9.67938931298
()
------------------------------
Acc X2 : -3.56017786865
Acc Y2 : -8.04930598145
Acc Z2 : 2.76051451416
()
Gyro X2 : 1.09160305344
Gyro Y2 : -1.06106870229
Gyro Z2 : -0.572519083969
()
------------------------------
knee_angle: 74
stepCount: 2
Temp : 27.8241176471
()
Acc X1 : -3.73256038818
Acc Y1 : 2.54982476807
Acc Z1 : 19.6127014496
()
Gyro X1 : 15.9083969466
Gyro Y1 : 209.267175573
Gyro Z1 : 2.75572519084
()
------------------------------
Acc X2 : -3.60806190186
Acc Y2 : -8.10197841797
Acc Z2 : 2.79163913574
()
Gyro X2 : 1.7786259542
Gyro Y2 : -1.42748091603
Gyro Z2 : -0.595419847328
()
```

geany_run_script_A3K420.sh

File Edit Tabs Help

```
knee_angle: 87
stepCount: 2
Temp : 27.73
()
Acc X1 : -1.76931502686
Acc Y1 : 2.50912333984
Acc Z1 : 12.1170546021
()
Gyro X1 : 17.7251908397
Gyro Y1 : 184.610687023
Gyro Z1 : 61.0305343511
()
--------------------------------
Acc X2 : -3.62003291016
Acc Y2 : -8.09240161133
Acc Z2 : 2.83952316895
()
Gyro X2 : 1.32824427481
Gyro Y2 : -0.93893129771
Gyro Z2 : -0.69465648855
()
--------------------------------
knee_angle: 98
stepCount: 3
Temp : 27.8241176471
()
Acc X1 : 10.2088758789
Acc Y1 : 1.75973822021
Acc Z1 : -1.2378022583
()
Gyro X1 : 89.2824427481
Gyro Y1 : 156.450381679
Gyro Z1 : 62.5877862595
()
--------------------------------
Acc X2 : -3.59369669189
Acc Y2 : -8.05409438477
Acc Z2 : 2.72460148926
()
Gyro X2 : 1.18320610687
Gyro Y2 : -0.954198473282
Gyro Z2 : -0.587786259542
()
--------------------------------
knee_angle: 99
stepCount: 3
Temp : 27.8711764706
()
Acc X1 : -4.09408483887
Acc Y1 : 3.39258375244
Acc Z1 : 13.5799118164
()
Gyro X1 : 152.007633588
Gyro Y1 : 250.129770992
Gyro Z1 : 93.1832061069
()
--------------------------------
Acc X2 : -3.58651408691
Acc Y2 : -7.97508572998
Acc Z2 : 2.86107098389
()
Gyro X2 : 1.29007633588
Gyro Y2 : -0.923664122137
Gyro Z2 : -0.671755725191
()
--------------------------------
```

geany_run_script_GGP620.sh

File Edit Tabs Help

```
------------------------------
knee_angle: 81
stepCount: 4
Temp : 29.3770588235
()
Acc X1 : 4.656722229
Acc Y1 : 0.0191536132812
Acc Z1 : 7.9798741333
()
Gyro X1 : -0.977099236641
Gyro Y1 : 0.114503816794
Gyro Z1 : -0.549618320611
()
------------------------------
Acc X2 : -2.4636335083
Acc Y2 : -8.36055219727
Acc Z2 : 2.99754047852
()
Gyro X2 : 1.24427480916
Gyro Y2 : -1.02290076336
Gyro Z2 : -0.648854961832
()
------------------------------
knee_angle: 42
stepCount: 5
Temp : 28.9064705882
()
Acc X1 : -5.26245524902
Acc Y1 : -0.122104284668
Acc Z1 : 7.04134708252
()
Gyro X1 : -0.93893129771
Gyro Y1 : 0.793893129771
Gyro Z1 : -0.312977099237
()
------------------------------
Acc X2 : 0.0
Acc Y2 : -7.62792648926
Acc Z2 : 4.90571920166
()
Gyro X2 : 1.0
Gyro Y2 : -0.923664122137
Gyro Z2 : -0.740458015267
()
------------------------------
knee_angle: 59
stepCount: 5
Temp : 28.1535294118
()
Acc X1 : -5.34625230713
Acc Y1 : -0.076614453125
Acc Z1 : 6.9767036377
()
Gyro X1 : -0.984732824427
Gyro Y1 : 0.916030534351
Gyro Z1 : -0.0458015267176
()
------------------------------
Acc X2 : 0.0263362182617
Acc Y2 : -7.6494743042
Acc Z2 : 4.79558592529
()
Gyro X2 : 1.09160305344
Gyro Y2 : -0.931297709924
Gyro Z2 : -0.625954198473
()
```

geany_run_script_L7U320.sh

File Edit Tabs Help

```
--------------------------------
knee_angle: 62
stepCount: 6
Temp : 28.1535294118
()
Acc X1 : -5.30555087891
Acc Y1 : -0.0957680664062
Acc Z1 : 7.01740506592
()
Gyro X1 : -0.961832061069
Gyro Y1 : 0.793893129771
Gyro Z1 : -0.0534351145038
()
--------------------------------
Acc X2 : 0.0670376464844
Acc Y2 : -7.61595548096
Acc Z2 : 4.93684382324
()
Gyro X2 : 1.29007633588
Gyro Y2 : -0.786259541985
Gyro Z2 : -0.63358778626
()
--------------------------------
knee_angle: 85
stepCount: 7
Temp : 28.2005882353
()
Acc X1 : -5.34146390381
Acc Y1 : -0.0957680664062
Acc Z1 : 7.08683691406
()
Gyro X1 : -0.946564885496
Gyro Y1 : 0.824427480916
Gyro Z1 : -0.335877862595
()
--------------------------------
Acc X2 : 0.0478840332031
Acc Y2 : -7.64229169922
Acc Z2 : 4.92247861328
()
Gyro X2 : 1.09160305344
Gyro Y2 : -0.908396946565
Gyro Z2 : -0.893129770992
()
--------------------------------
knee_angle: 76
stepCount: 7
Temp : 29.9888235294
()
Acc X1 : 3.36624753418
Acc Y1 : -1.64002813721
Acc Z1 : 13.9390420654
()
Gyro X1 : -46.534351145
Gyro Y1 : -131.160305344
Gyro Z1 : -8.95419847328
()
--------------------------------
Acc X2 : -3.65594593506
Acc Y2 : -8.07564219971
Acc Z2 : 2.77966812744
()
Gyro X2 : 1.36641221374
Gyro Y2 : -0.954198473282
Gyro Z2 : -0.748091603053
()
--------------------------------
```