**CHAPTER 1**

**INTRODUCTION**

* 1. **Motivation**

In today’s world technology advancement is going in a huge speed so as we all. But still there are many fields which needs to be automated through time. Such one example of an industry which needs to be modernized in terms of technology and that is healthcare industry.

As in present time every device which is going to come on the market and the maximum devices which are already present in the market has internet connectivity. All devices around us from a small household item to a big machinery everything is becoming smart. So, the healthcare industry should also become smart in order to give better service to the customers.

In this field a huge amount of research and development is going on all over the world, small startup companies are trying to build a solution for a current problem statement with the help of the current technology Internet of Things.

* 1. **Purpose**

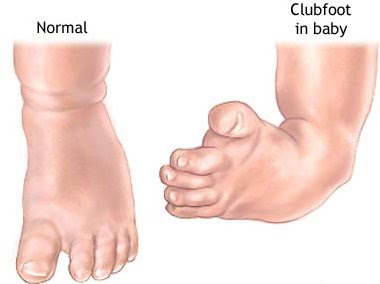
The main purpose of this device is to shorten the gap between old devices and the modern day technology. As in any industry there are many types of devices solving different types of problem areas but as time is moving on the technology advancement is also happening and these devices are left behind so the motive is to bring these devices to the main stream which is possible through connecting those devices online and use them as new Iot devices for related fields.

* 1. **Scope**

There is a disease known as clubfoot which effects the new born baby. A doctor named Ignacio Ponseti who discovered a shoe which is used to cure the patient by following some instructions given by the doctor.

* 1. **Literature survey**

Mean follow-up was 69 months. Нe total numbers of Ponseti casts before tenotomy, details of the tenotomy, and compliance with the CTEV brace were recorded. On average, 6.6 (range 4-13) casts were necessary before performing the tenotomy. Tenotomy was carried out by a single surgeon (V.P.) in a total of 155 feet (74.9%), always performed in an operating room, with the patient under general anesthesia, by a percutaneous approach. Achilles tenotomy was performed aіer the foot had been abducted to at least 60° and when there were less than 10° of dorsLflexLon. Нe mean post-operative Pirani score was 0.35, showing good/excellent results in 194 (93.7%) feet (Figures 1A-1D). Only 5 patients (3.8%), 8 clubfeet (3.9%), relapsed. Poor compliance with the Denis Browne splint was thought to be the main cause of failure (Table Нese fiJures show the dLوٴerent phases of the CTEV treatment applying the Ponseti Method. 1A. Congenital talipes equinovarus, Pirani’s score 5.5; 1B. Ponseti’s cast treatment; 1C: Achilles tenotomy; 1D: Denis-Browne splint. Discussion CTEV is the most common abnormality of the musculoskeletal system. Нe incidence of CTEV is between 1 and 2 per 1000 live births within the Caucasian population, but there are sLJnLficant racial variations; CTEV is more common in the Polynesian population and less common in the Chinese [1,2,10]. Нe incidence of CTEV in Sicily was previously reported by Pavone among the 801,324 newborns recorded in Sicily between January1991 and December 2004, the prevalence of clubfoot was 1.03 per 1000 births, in accord with the most recent data from the literature. 'Lوٴerent genetic, developmental, and environmental factors predispose the etiopathogenesis of congenital clubfoot. Family studies demonstrated a higher prevalence among first-deJree relatives and a concordance of 33% among monozygotic twins, which decreased to 3% for dizygotic twins. Genetic susceptibility can also explain the double incidence of CTEV among males in comparison to females . A five-JeneratLon family with members aوٴected by asymmetrical right-sided predominant idiopathic CTEV with autosomal dominant inheritance with incomplete penetrance was reported by Gurnett. In the same family, other members showed other limb malformations, including patellar hypoplasia, oblique talus, tibia hemimelia, and preaxial polydactyly. In this family, a single missense mutation in PITX-1, a bicoid-related homeodomain transcription factor involved in limb formation, played a relevant role in the pathogenesis of these limb abnormalities. Нe genes involved in congenital joint contractures characterize specLfic syndromes, including distal arthrogryposis. Moreover, anomalies in chromosomal deletions and duplications have been thought to be responsible for CTEV. RNA binding motif protein 10 (RBM-10) mutations were related to TARP syndrome, in which the patient shows talipes equinovarus in association with an atrial septal defect and Robin sequence.

This device has some flaws. There was a lack of proper following of the prescript instruction by the parents and there was not a proper communication between the doctor and the patient which led to carelessness of parents which further leads to reduction of success rate of the device. This was the main problem in the device. One more issue was as these devices have a huge technological gap i.e. the disease remained the same the device also remained the same till date. There was no data kept in record for the device, which is very useful in many terms as that data can be processed to predict the cause of the disease and to find a cure for it and put an end to this disease. This was also a problem in that device.

So, the product which we have developed is solving all those major problems which were responsible for the reduction of success rate of the device. There are 2 Fsr pressure sensors placed on board which are going to give readings about the amount of pressure applied on the foot of a patient. There is a onboard led and buzzer which are the alarming system of the device. They are triggered when the foot of the patient is not placed properly or the straps of the shoes are not tied properly, so the parents will get to know which leg is facing issues as there are two buzzers and led for both the legs.

The doctor will able to keep a track on the patient whether the prescribed time table is been followed or not and the data will also be stored in the cloud which can be used for different other applications or data analytics and machine learning can be applied to predict for the future cases so safety measures can be taken for it.

**CHAPTER 2:**

**SOFTWARE REQUIREMENTS**

**2.1 Product Perspective**

The product is based on Internet of Things. We have developed a solution that is going to monitor patient’s data. The platform on which the product is being developed is an open source, under the GNU General Public License. We are connecting patient and doctor through our product for a particular type of disease known as Clubfoot. The product is automating the healthcare industry.

The following are the main features that are included in our product.

* On device Wi-Fi module: Offers connectivity of the device globally.
* Real Time Monitoring: User can store and access the data at real time.
* One Time Initialization: User only have to set the device once.
* Low Cost: Production and Automation cost is affordable.
* High Efficiency: This product can solve the biggest problem which is faced while treatment of clubfoot is carelessness of parents which is being removed completely by our product hence increasing the rate of success and the efficiency of the device is increased.

**2.2 User Characteristics**

The user only need a basic knowledge of browsing internet. The doctor need to have a basic knowledge of the product, how it works, and how to use which will be provided in the product manual and the important thing is it is very simple to use.

**2.3 Constraints**

Every product has some constraints but this one is having very less such as:

* The device should be connected to internet always to perform much better.
* The batteries must be replaced or recharged as they get to discharge in around 1-3 days.

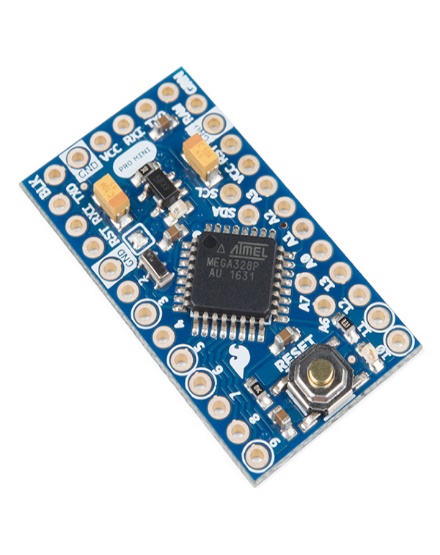
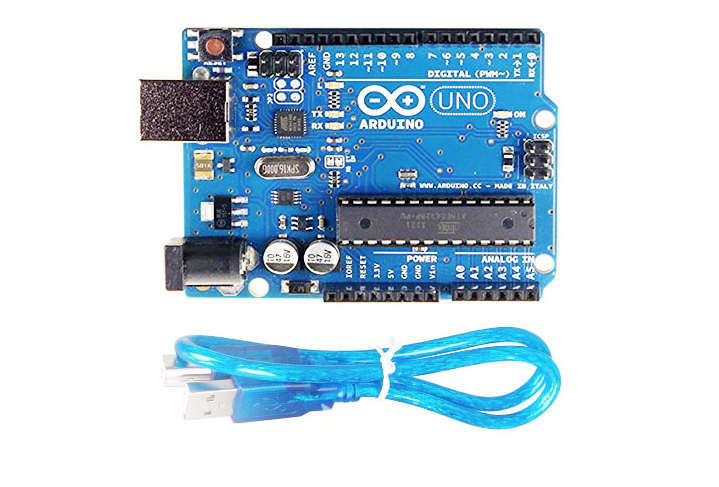
**2.4 Assumption and Dependencies**

In future we are going to collect a huge number of data from patients all across the world and we will apply machine learning algorithms and data analytics to reduce the number of people suffering from clubfoot. This product highly depends on internet connectivity as it is Internet of Things product or we can say as smart healthcare devices.

**2.5 Specific Requirements**

**2.5.1 HARDWARE:**

* **ARDUINO UNO/pro Mini**

****

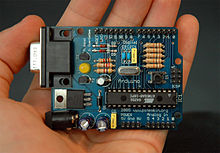
Arduino is an open-source electronics platform based on easy-to-use hardware and software. [Arduino boards](https://www.arduino.cc/en/Main/Products) are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the [Arduino programming language](https://www.arduino.cc/en/Reference/HomePage) (based on [Wiring](http://wiring.org.co/)), and [the Arduino Software (IDE)](https://www.arduino.cc/en/Main/Software), based on [Processing](https://processing.org/).

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of [accessible knowledge](http://forum.arduino.cc/) that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The [software](https://www.arduino.cc/en/Main/Software), too, is open-source, and it is growing through the contributions of users worldwide.

Arduino is [open-source hardware](https://en.wikipedia.org/wiki/Open-source_hardware). The hardware reference designs are distributed under a [Creative Commons](https://en.wikipedia.org/wiki/Creative_Commons) Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available.

Although the hardware and software designs are freely available under [copyleft](https://en.wikipedia.org/wiki/Copyleft) licenses, the developers have requested the name *Arduino*to be [exclusive to the official product](https://en.wikipedia.org/wiki/Generic_trademark) and not be used for derived works without permission. The official policy document on use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product.[[21]](https://en.wikipedia.org/wiki/Arduino#cite_note-AutoF7-44-21) Several Arduino-compatible products commercially released have avoided the project name by using various names ending in *-duino*.[[22]](https://en.wikipedia.org/wiki/Arduino#cite_note-freeduino-22)

[](https://en.wikipedia.org/wiki/File:Arduino316.jpg)

An early Arduino board[[23]](https://en.wikipedia.org/wiki/Arduino#cite_note-23) with an [RS-232](https://en.wikipedia.org/wiki/RS-232) [serial](https://en.wikipedia.org/wiki/Serial_communication) interface (upper left) and an Atmel ATmega8 microcontroller chip (black, lower right); the 14 digital I/O pins are at the top, the 6 analog input pins at the lower right, and the power connector at the lower left.

Most Arduino boards consist of an [Atmel](https://en.wikipedia.org/wiki/Atmel) 8-bit [AVR microcontroller](https://en.wikipedia.org/wiki/AVR_microcontroller) (ATmega8,[[24]](https://en.wikipedia.org/wiki/Arduino#cite_note-24) ATmega168, [ATmega328](https://en.wikipedia.org/wiki/ATmega328), ATmega1280, ATmega2560) with varying amounts of flash memory, pins, and features.[[25]](https://en.wikipedia.org/wiki/Arduino#cite_note-25) The 32-bit [Arduino Due](https://en.wikipedia.org/wiki/Arduino_Due), based on the Atmel [SAM3X8E](https://en.wikipedia.org/wiki/Atmel_ARM-based_processors#SAM_3) was introduced in 2012.[[26]](https://en.wikipedia.org/wiki/Arduino#cite_note-26) The boards use single or double-row pins or female headers that facilitate connections for programming and incorporation into other circuits. These may connect with add-on modules termed *shields*. Multiple and possibly stacked shields may be individually addressable via an [I²C](https://en.wikipedia.org/wiki/I%C2%B2C) [serial bus](https://en.wikipedia.org/wiki/Serial_bus). Most boards include a 5 V [linear regulator](https://en.wikipedia.org/wiki/Linear_regulator) and a 16 MHz [crystal oscillator](https://en.wikipedia.org/wiki/Crystal_oscillator) or [ceramic resonator](https://en.wikipedia.org/wiki/Ceramic_resonator). Some designs, such as the LilyPad, run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions.

Arduino microcontrollers are pre-programmed with a [boot loader](https://en.wikipedia.org/wiki/Boot_loader) that simplifies uploading of programs to the on-chip [flash memory](https://en.wikipedia.org/wiki/Flash_memory). The default bootloader of the Arduino UNO is the optiboot bootloader.[[27]](https://en.wikipedia.org/wiki/Arduino#cite_note-27) Boards are loaded with program code via a serial connection to another computer. Some serial Arduino boards contain a level shifter circuit to convert between [RS-232](https://en.wikipedia.org/wiki/RS-232) logic levels and [transistor–transistor logic](https://en.wikipedia.org/wiki/Transistor%E2%80%93transistor_logic)(TTL) level signals. Current Arduino boards are programmed via [Universal Serial Bus](https://en.wikipedia.org/wiki/Universal_Serial_Bus) (USB), implemented using USB-to-serial adapter chips such as the [FTDI](https://en.wikipedia.org/wiki/FTDI) FT232. Some boards, such as later-model Uno boards, substitute the [FTDI](https://en.wikipedia.org/wiki/FTDI) chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own [ICSP](https://en.wikipedia.org/wiki/In-system_programming) header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, [Bluetooth](https://en.wikipedia.org/wiki/Bluetooth) or other methods. When used with traditional microcontroller tools, instead of the Arduino IDE, standard AVR [in-system programming](https://en.wikipedia.org/wiki/In-system_programming) (ISP) programming is used.

[](https://en.wikipedia.org/wiki/File:UnoConnections.jpg)

An official Arduino Uno R2 with descriptions of the I/O locations

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The *Diecimila*, *Duemilanove*, and current *Uno* provide 14 digital I/O pins, six of which can produce [pulse-width modulated](https://en.wikipedia.org/wiki/Pulse-width_modulation) signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Boardand Boarduinoboards may provide male header pins on the underside of the board that can plug into solderless [breadboards](https://en.wikipedia.org/wiki/Breadboard).

Many Arduino-compatible and Arduino-derived boards exist. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use different processors, of varying compatibility.

* **FSR Pressure Sensor**

A pressure sensor is a device for [pressure measurement](https://en.wikipedia.org/wiki/Pressure_measurement) of [gases](https://en.wikipedia.org/wiki/Gas) or [liquids](https://en.wikipedia.org/wiki/Liquids). Pressure is an expression of the force required to stop a fluid from expanding, and is usually stated in terms of force per unit area. A pressure sensor usually acts as a [transducer](https://en.wikipedia.org/wiki/Transducer); it generates a signal as a [function](https://en.wikipedia.org/wiki/Function_(mathematics)) of the pressure imposed. For the purposes of this article, such a signal is electrical.

Pressure sensors are used for control and monitoring in thousands of everyday applications. Pressure sensors can also be used to indirectly measure other variables such as fluid/gas flow, speed, [water level](https://en.wikipedia.org/wiki/Water_level), and [altitude](https://en.wikipedia.org/wiki/Altitude).

Pressure sensors can vary drastically in technology, design, performance, application suitability and cost. A conservative estimate would be that there may be over 50 technologies and at least 300 companies making pressure sensors worldwide.

There is also a category of pressure sensors that are designed to measure in a dynamic mode for capturing very high speed changes in pressure. Example applications for this type of sensor would be in the measuring of combustion pressure in an engine cylinder or in a gas turbine. These sensors are commonly manufactured out of [piezoelectric](https://en.wikipedia.org/wiki/Piezoelectric) materials such as quartz.

Some pressure sensors are [pressure switches](https://en.wikipedia.org/wiki/Pressure_switch), which turn on or off at a particular pressure. For example, a water pump can be controlled by a pressure switch so that it starts when water is released from the system, reducing the pressure in a reservoir.

Pressure sensors can be classified in terms of pressure ranges they measure, temperature ranges of operation, and most importantly the type of pressure they measure. Pressure sensors are variously named according to their purpose, but the same technology may be used under different names.

* **Absolute pressure sensor**

This sensor measures the pressure relative to [perfect vacuum](https://en.wikipedia.org/wiki/Vacuum).

* **Gauge pressure sensor**

This sensor measures the pressure relative to [atmospheric pressure](https://en.wikipedia.org/wiki/Atmospheric_pressure). A tire pressure gauge is an example of gauge pressure measurement; when it indicates zero, then the pressure it is measuring is the same as the ambient pressure.

* **Vacuum pressure sensor**

This term can cause confusion. It may be used to describe a sensor that measures pressures below atmospheric pressure, showing the difference between that low pressure and atmospheric pressure, but it may also be used to describe a sensor that measures absolute pressure relative to a vacuum.

* **Differential pressure sensor**

This sensor measures the difference between two pressures, one connected to each side of the sensor. Differential pressure sensors are used to measure many properties, such as pressure drops across [oil filters](https://en.wikipedia.org/wiki/Oil_filter) or [air filters](https://en.wikipedia.org/wiki/Air_filter), fluid levels (by comparing the pressure above and below the liquid) or flow rates (by measuring the change in pressure across a restriction). Technically speaking, most pressure sensors are really differential pressure sensors; for example a gauge pressure sensor is merely a differential pressure sensor in which one side is open to the ambient atmosphere.

* **Sealed pressure sensor**

This sensor is similar to a gauge pressure sensor except that it measures pressure relative to some fixed pressure rather than the ambient atmospheric pressure (which varies according to the location and the weather).

****

* **ESP8266 Wifi Module:**

The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by manufacturer Espressif Systems[1] in Shanghai, China.

The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. However, at first there was almost no English-language documentation on the chip and the commands it accepted. The very low price and the fact that there were very few external components on the module, which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, chip, and the software on it, as well as to translate the Chinese documentation.

The ESP8285 is an ESP8266 with 1 MiB of built-in flash, allowing for single-chip devices capable of connecting to Wi-Fi.

The successor to these microcontroller chips is the ESP32, released in 2016.

The DeviceBit platform (http://www.devicebit.com/home/publicsensors) is a real-time data brokerage platform for the Internet of Things (IoT), providing most of its functionality via its Application Programming Interface (API). It is quick and easy to add Devices and Applications to the DeviceBit platform. It provides real-time data storage and remote control at scale. The DeviceBit platform is not just an easy way to prototype new Internet-enabled sensors; it's also a service that helps companies bring products to market at scale.

The DeviceBit platform provides basic data analysis tools for rapid data evaluation, as well as real-time alerts and notifications if sensors report “abnormal” conditions.

You can realize your own ideas and develop your own devices relying on this platform. You can focus on hardware instead of software infrastructure.

The DeviceBit platform also communicates with existing social network, such as Twitter and Facebook, allowing you to share what you do with your friend, which might be very helpful to their research on similar field.

****

* **Ponseti Shoes:**

The Ponseti method is a manipulative technique that corrects congenital clubfoot without invasive surgery. It was developed by Ignacio V. Ponseti of the University of Iowa Hospitals and Clinics, USA in the 1950s, and was repopularized in 2000 by John Herzenberg in the USA and Europe and in Africa by NHS surgeon Steve Mannion. It is a standard treatment for club foot.

Ponseti treatment was introduced in UK in the late 1990s and widely popularized around the country by NHS physiotherapist Steve Wildon. The manipulative treatment of clubfoot deformity is based on the inherent properties of the connective tissue, cartilage, and bone, which respond to the proper mechanical stimuli created by the gradual reduction of the deformity. The ligaments, joint capsules, and tendons are stretched under gentle manipulations. A plaster cast is applied after each manipulation to retain the degree of correction and soften the ligaments. The displaced bones are thus gradually brought into the correct alignment with their joint surfaces progressively remodeled yet maintaining congruency. After two months of manipulation and casting the foot appears slightly over-corrected. After a few weeks in splints however, the foot looks normal.

Proper foot manipulations require a thorough understanding of the anatomy and kinematics of the normal foot and of the deviations of the tarsal bones in the clubfoot. Poorly conducted manipulations will further complicate the clubfoot deformity. The non-operative treatment will succeed better if it is started a few days or weeks after birth and if the podiatrist understands the nature of the deformity and possesses manipulative skill and expertise in plaster-cast applications.[1]

The Ponseti's technique is painless, fast, cost-effective and successful in almost 100% of all congenital clubfoot cases. The Ponseti method is endorsed and supported by World Health Organization [2][3] National Institutes of Health,[4] American Academy of Orthopedic Surgeons,[5] Pediatric Orthopedic Society of North America,[6] European Pediatric Orthopedic Society,[7] CURE International,[8] STEPS Charity UK,[9] STEPS Charity South Africa,[10] and A Leg to Stand On (India).[11]

1. The calcaneal internal rotation (adduction) and plantar flexion is the key deformity. The foot is adducted and plantar-flexed at the subtalar joint, and the goal is to abduct the foot and dorsiflex it. In order to achieve correction of the clubfoot, the calcaneus should be allowed to rotate freely under the talus bone, which also is free to rotate in the ankle mortise. The correction takes place through the normal arc of the subtalar joint. This is achieved by placing the index finger of the operator on the medial malleolus to stabilize the leg and levering on the thumb placed on the lateral aspect head of the talus while abducting the forefoot in supination. Forcible attempts at correcting the heel varus by abducting the forefoot while applying counter pressure at the calcaneocuboid joint prevents the calcaneus from abducting and therefore everting.

2. Foot cavus increases when the forefoot is pronated. If cavus is present, the first step in the manipulation process is to supinate the forefoot by gently lifting the dropped first metatarsal to correct the cavus. Once the cavus is corrected, the forefoot can be abducted as outlined in step 1.

3. Pronation of the foot also causes the calcaneus to jam under the talus. The calcaneum cannot rotate and stays in varus. The cavus increases as outlined in step 2. This results in a bean-shaped foot. At the end of step 1, the foot is maximally abducted but never pronated.

4. The manipulation is carried out in the cast room, with the baby having been fed just prior to the treatment or even during the treatment. After the foot is manipulated, a long leg cast is applied to hold the correction. Initially, the short leg component is applied. The cast should be snug with minimal but adequate padding. The authors paint or spray the limb with tincture of benzoin to allow adherence of the padding to the limb. The authors prefer to apply additional padding strips along the medial and lateral borders to facilitate safe removal of the cast with a cast saw. The cast must incorporate the toes right up to the tips but not squeeze the toes or obliterate the transverse arch. The cast is molded to contour around the heel while abducting the forefoot against counter pressure on the lateral aspect of the head of the talus. The knee is flexed to 90° for the long leg component of the cast. The parents can soak these casts for 30–45 minutes prior to removal with a plaster knife. The authors' preferred method is to use the oscillating plaster saw for cast removal. The cast is bivalved and removed. The cast then is reconstituted by coapting the two halves. This allows for monitoring of the progress of the forefoot abduction and, in the later stages, the amount of dorsiflexion or equinus correction.

5. Forcible correction of the equinus (and cavus) by dorsiflexion against a tight Achilles tendon results in a spurious correction through a break in the midfoot, resulting in a rocker-bottom foot. The cavus should be separately treated as outlined in step 2, and the equinus should be corrected without causing a midfoot break. It generally takes up to 4–7 casts to achieve maximum foot abduction. The casts are changed weekly. The foot abduction (correction) can be considered adequate when the thigh-foot axis is 60°. After maximal foot abduction is obtained, most cases require a percutaneous Achilles tenotomy. This is performed in the cast room under aseptic conditions. The local area is anesthetized with a combination of a topical lignocaine preparation (e.g. EMLA cream) and minimal local infiltration of lidocaine. The tenotomy is performed through a stab incision with a round tip (#6400) Beaver blade. The wound is closed with a single absorbable suture or with adhesive strips.The final cast is applied with the foot in maximum dorsiflexion, and the foot is held in the cast for 2–3 weeks.

6. Following the manipulation and casting phase, the feet are fitted with open-toed straight-laced shoes attached to a Denis Browne bar. The affected foot is abducted (externally rotated) to 70° with the unaffected foot set at 45° of abduction. The shoes also have a heel counter bumper to prevent the heel from slipping out of the shoe. The shoes are worn for 23 hours a day for three months and are worn at night and during naps for up to three years.

7. In 10–30% of cases, a tibialis anterior tendon transfer to the lateral cuneiform is performed when the child is approximately three years of age. This gives lasting correction of the forefoot, preventing metatarsus adductus and foot inversion. This procedure is indicated in a child aged 2–2.5 years with dynamic supination of the foot. Prior to surgery, cast the foot in a long leg cast for a few weeks to regain the correction.

****

* **Buzzer:**

A buzzer or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

The electric buzzer was invented in 1831 by [Joseph Henry](https://en.wikipedia.org/wiki/Joseph_Henry). They were mainly used in early [doorbells](https://en.wikipedia.org/wiki/Doorbell) until they were phased out in the early 1930s in favor of musical chimes, which had a softer tone.[[2]](https://en.wikipedia.org/wiki/Buzzer#cite_note-2)

[Piezoelectric](https://en.wikipedia.org/wiki/Piezoelectricity) buzzers, or piezo buzzers, as they are sometimes called, were invented by Japanese manufacturers and fitted into a wide array of products during the 1970s to 1980s. This advancement mainly came about because of cooperative efforts by Japanese manufacturing companies. In 1951, they established the Barium Titanate Application Research Committee, which allowed the companies to be "competitively cooperative" and bring about several piezoelectric innovations and inventions.

Early devices were based on an electromechanical system identical to an [electric bell](https://en.wikipedia.org/wiki/Electric_bell) without the metal gong. Similarly, a [relay](https://en.wikipedia.org/wiki/Relay) may be connected to interrupt its own actuating [current](https://en.wikipedia.org/wiki/Electric_current), causing the [contacts](https://en.wikipedia.org/wiki/Switch) to buzz. Often these units were anchored to a wall or ceiling to use it as a sounding board. The word "buzzer" comes from the rasping noise that electromechanical buzzers made.

Many alarm clocks have [radio receivers](https://en.wikipedia.org/wiki/Radio_receiver) that can be set to start playing at specified times, and are known as *clock radios*. Some alarm clocks can set multiple alarms. A *progressive alarm clock*, can have different alarms for different times (see [Next-Generation Alarms](https://en.wikipedia.org/wiki/Alarm_clock#Next-generation_alarms)) and even play music of your choice. Most modern televisions, mobile phones and digital watches have alarm clock functions to turn on or make sounds at a specific time.

****

**2.5.2 SOFTWARE:**

Ide: Arduino (any version)

Operating System: Windows, Mac, Linux

RAM: 512 mb

**2.5.3 Other requirements:**

The doctor should make an account on thingspeak to monitor the patient’s data and to keep a record of that data**.**

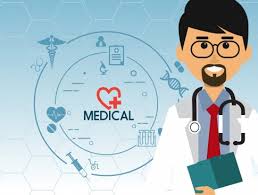
* Account in thingspeak (an open source cloud platform for IoT devices).
* Press a button present on the shoe to trigger the alarming function of the device.

**CHAPTER 3**

**DESIGN**



cloud

 **** 

User Ponseti ShoesDoctor



Monitoring

Fig 3.1: System Architecture for Ponseti Shoes data flow

**3.1 CLOUD:**

Cloud computing is the on-demand availability of [computer](https://en.wikipedia.org/wiki/Computer) [system resources](https://en.wikipedia.org/wiki/System_resource), especially [data storage](https://en.wikipedia.org/wiki/Data_storage) and [computing power](https://en.wikipedia.org/wiki/Computing_power), without direct active management by the user. The term is generally used to describe data centers available to many users over the [Internet](https://en.wikipedia.org/wiki/Internet). Large clouds, predominant today, often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close, it may be designated an [edge server](https://en.wikipedia.org/wiki/Edge_server).

Clouds may be limited to a single organization (enterprise clouds[[1]](https://en.wikipedia.org/wiki/Cloud_computing#cite_note-1)[[2]](https://en.wikipedia.org/wiki/Cloud_computing#cite_note-aws.amazon-2),) be available to many organizations (public cloud,) or a combination of both ([hybrid cloud](https://en.wikipedia.org/wiki/Hybrid_Cloud_Gateway)).

Cloud computing relies on sharing of resources to achieve coherence and [economies of scale](https://en.wikipedia.org/wiki/Economies_of_scale).

Advocates of public and hybrid clouds note that cloud computing allows companies to avoid or minimize up-front [IT infrastructure](https://en.wikipedia.org/wiki/IT_infrastructure) costs. Proponents also claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and that it enables IT teams to more rapidly adjust resources to meet fluctuating and unpredictable demand.[[2]](https://en.wikipedia.org/wiki/Cloud_computing#cite_note-aws.amazon-2)[[3]](https://en.wikipedia.org/wiki/Cloud_computing#cite_note-3)[[4]](https://en.wikipedia.org/wiki/Cloud_computing#cite_note-4)Cloud providers typically use a "pay-as-you-go" model, which can lead to unexpected [operating expenses](https://en.wikipedia.org/wiki/Operating_expense) if administrators are not familiarized with cloud-pricing models.[[5]](https://en.wikipedia.org/wiki/Cloud_computing#cite_note-5)

The availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of [hardware virtualization](https://en.wikipedia.org/wiki/Hardware_virtualization), [service-oriented architecture](https://en.wikipedia.org/wiki/Service-oriented_architecture), and [autonomic](https://en.wikipedia.org/wiki/Autonomic_computing) and [utility computing](https://en.wikipedia.org/wiki/Utility_computing) has led to growth in cloud computing.[[6]](https://en.wikipedia.org/wiki/Cloud_computing#cite_note-6)[[7]](https://en.wikipedia.org/wiki/Cloud_computing#cite_note-gartner-7)[[8]](https://en.wikipedia.org/wiki/Cloud_computing#cite_note-really-8)

### Cloud Computing Services Features and Benefits

* Hosted and maintained by the provider. The [cloud hosting](https://www.akamai.com/us/en/resources/cloud-hosting.jsp) provider purchases, hosts, and maintains the necessary hardware and software in their own facility. Service users avoid the capital expenditures and maintenance headaches that they would have if they developed the service themselves on-premise.
* Self-service through a web interface. Service users can initiate specific service functions, and increase or decrease their service usage level, though a web interface with little or no interaction with the service provider.
* Pay for use. Service users pay only for the amount of service that they use. This can result in substantial cost savings compared to the traditional approach of developing on-site IT capacities geared toward maximum usage scenarios, and then having that capacity be under-utilized much of the time.
* Near-limitless scalability. Cloud computing services providers typically have the infrastructure to deliver their service at massive scale. For cloud service users, that means that the cloud can easily accommodate business growth or periodic spikes in service usage.

### Cloud Computing Services Types

* Infrastructure as a Service (IaaS). IaaS provides users access to raw computing resources such processing power, data storage capacity, and networking, in the context of a secure data center.
* Platform as a Service (PaaS). Geared toward software development teams, PaaS offerings provide computing and storage infrastructure and also a development platform layer, with components such as web servers, database management systems, and software development kits (SDKs) for various programming languages.
* Software as a Service (SaaS). SaaS providers offer application-level services tailored to a wide variety of business needs, such as customer relationship management (CRM), marketing automation, or business analytics.

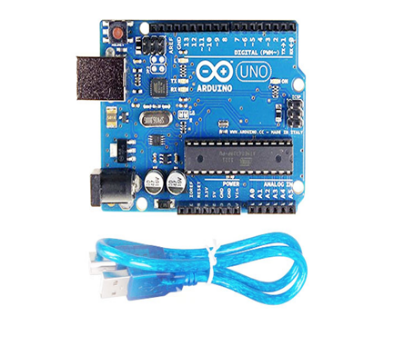
### Cloud Computing Services Acceleration.

**3.2** **Data Flow Diagram:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

PSR sensor Monitor

Processing Receive Data Send data 

Wifi Module Cloud

Fig 3.1:

**3.3** **UML Diagram**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Numerous things become possible through the use of UML. One can run [network](https://en.wikipedia.org/wiki/Computer_network) services from a UML environment and remain totally sequestered from the main Linux system in which the UML environment runs. Administrators can use UML to set up [honeypots](https://en.wikipedia.org/wiki/Honeypot_(computing)), which allow one to test the security of one's computers or network. UML can serve to test and debug new software without adversely affecting the host system. UML can also be used for teaching and research, providing a realistic Linux networked environment with a high degree of safety.

In UML environments, host and guest kernel versions don't need to match, so it is entirely possible to test a "[bleeding edge](https://en.wikipedia.org/wiki/Bleeding_edge)" version of Linux in User-mode on a system running a much older kernel. UML also allows kernel debugging to be performed on one machine, where other kernel debugging tools (such as [kgdb](https://en.wikipedia.org/wiki/Kgdb" \o "Kgdb)) require two machines connected with a [null modem](https://en.wikipedia.org/wiki/Null_modem) cable.

Some [web hosting](https://en.wikipedia.org/wiki/Web_hosting) providers offer UML-powered [virtual servers](https://en.wikipedia.org/wiki/Virtual_private_server) for lower prices than true [dedicated servers](https://en.wikipedia.org/wiki/Dedicated_hosting_service). Each customer has [root](https://en.wikipedia.org/wiki/Root_user) access on what appears to be their own system, while in reality one physical computer is shared between many people.

[libguestfs](https://en.wikipedia.org/wiki/Libguestfs) has supported a UML backend since version 1.24 as an alternative to using QEMU or KVM.

**3.3.1** **Goals:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**3.4** **Use Case Diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

**3.5** **Input Design**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

**3.6** **Output Design**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**CHAPTER 4**

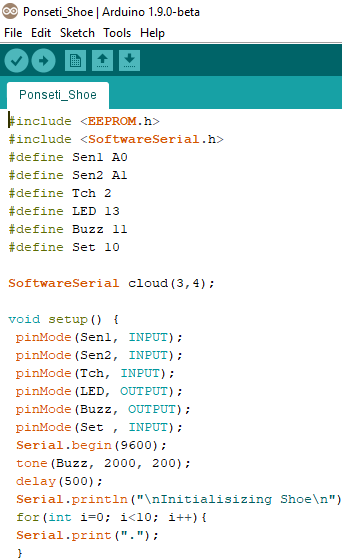
**IMPLEMENTATION**

**4.1 Implementation details:**

The implementation of the product is done in the following’s ways.

* First, we have got all the hardware components and should crosscheck whether the components are right ones or not.
* Next, we have to check whether all the hardware is working or not by running demo tests.
* After that we have to assemble all the components according to our requirement, i.e.: integrating all the analog sensors and the Wi-Fi module together with the development board.
* Next, we have to switch towards the software part where we have to program the board to further perform and to give the desired output.
* Now we will write the embedded c code to program our micro-controller (Atmega328p).
* After that we will try to connect the Wi-Fi module to the internet by adding few AT commands in the code.

**4.2 Complexity of program code:**

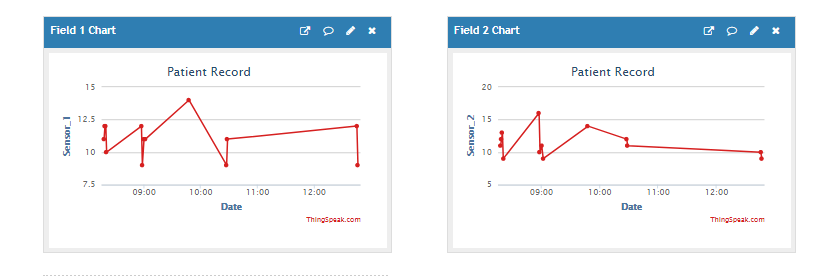


As we can see in the snapshot of the code that we have used embedded C programing language to program the board. To program Arduino, it is very simple. As it is an open source platform, developers from all around the world are contributing in it to make it simple for users to program it easily. The complexity of embedded programming is being reduced in Arduino by having a large set of pre-defined libraries for different modules and sensors which help user to integrate hardware and software easily by calling those functions in the program code.

As we can see in the snapshot that we have imported two header files, one is EPROM.h which is used to store the initial reading of the sensors in the eprom of the micro-controller which is further used for the alarming mechanism of the device.



As in the above screenshot we can see a little bit complexity in the code. In the above written code, we are integrating the esp8266 Wi-Fi module without cloud thinkspeak by using the API provided by thingspeak. We are uploading the sensors value to the cloud platform where it will be plotted on a graph for easy monitoring purpose.



**Chapter 5**

**TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising product with the intent of ensuring that the product meets its requirements and user expectations and does not fail in an unacceptable manner. The system has been verified and validated by running the test data and live data.

Testing is an investigation conducted to provide stakeholders with information about the [quality](https://en.wikipedia.org/wiki/Software_quality) of the [software](https://en.wikipedia.org/wiki/Software) product or service under test.[[1]](https://en.wikipedia.org/wiki/Software_testing#cite_note-Kaner_1-1) Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding [software bugs](https://en.wikipedia.org/wiki/Software_bug) (errors or other defects), and verifying that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

* meets the requirements that guided its design and development,
* responds correctly to all kinds of inputs,
* performs its functions within an acceptable time,
* is sufficiently usable,
* can be installed and run in its intended [environments](https://en.wikipedia.org/wiki/Operating_environment), and
* achieves the general result its stakeholders desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding [software bugs](https://en.wikipedia.org/wiki/Software_bug) (errors or other defects). The job of testing is an iterative process as when one bug is fixed, it can illuminate other, deeper bugs, or can even create new ones.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors.[[1]](https://en.wikipedia.org/wiki/Software_testing#cite_note-Kaner_1-1)

Software testing can be conducted as soon as executable software (even if partially complete) exists. The [overall approach to software development](https://en.wikipedia.org/wiki/Software_development_process) often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an [agile approach](https://en.wikipedia.org/wiki/Agile_software_development), requirements, programming, and testing are often done concurrently.

**5.1 Levels of Testing**

**5.1.1 Unit Testing**

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use.  Intuitively, one can view a unit as the smallest testable part of an application.  In object-oriented programming a unit is often an entire interface, such as a class, but could be an individual method.

For unit testing first we adopted the code testing strategy, which examined the logic of program. During the development process itself all the syntax errors etc. got rooted out. For this developed test case that result in executing every instruction in the program or module i.e. every path through program was tested. Test cases are data chosen at random to check every possible branch after all the loops.

**Test case (Arduino board):**

Table 5.1: Test cases for the Arduino board

|  |  |  |
| --- | --- | --- |
| **Steps** | **Test Action** | **Result** |
| 1 | Burn a simple program in to Arduino board. | Program burnt successfully. |
| 2 | Reset Arduino board. | - |
| 3 | Run program on Arduino. | Program run successfully. |

**Test case (Pressure sensor):**

Table 5.2: Test cases for the Pressure sensor

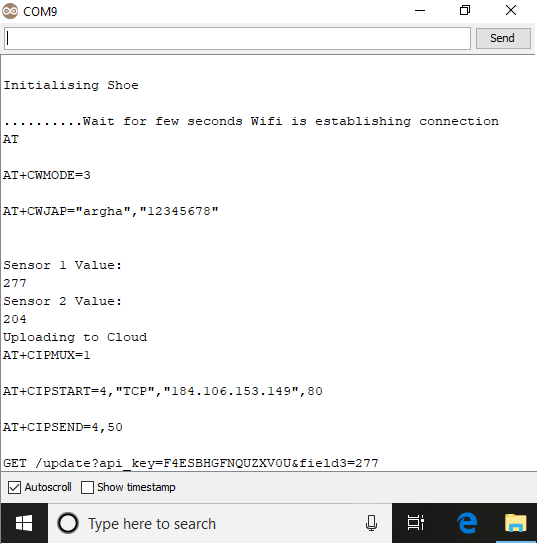
|  |  |  |
| --- | --- | --- |
| **Steps** | **Test Action** | **Result** |
| 1. | Burn a simple pressure sensor program in to Arduino board. | Program burnt successfully. |
| 2. | Reset Arduino board. | - |
| 3. | Connect pressure sensor to Arduino. | Connection successful. |
| 4. | Run program on Arduino, and apply pressure on pressure sensor at the same time | Buzzer make sound and input value is stored.  Pressure sensor working fine. |

**Test case (Wi-fi module):**

Table 5.1: Test cases for the wi-fi module

|  |  |  |
| --- | --- | --- |
| **Steps** | **Test Action** | **Result** |
| 1. | Connect wi-fi module with computer | Connected successfully. |
| 2. | Execute AT command in serial monitor | AT command executed successfully. |
| 3. | Burn a simple wi-fi module program in to Arduino board. | Program burnt successfully. |
| 4. | Reset Arduino board. | - |
| 5. | Connect wi-fi module to Arduino. | Connection successful. |
| 6. | Transmit data using wi-fi module. | Data transmit and receive successfully.  wi-fi module working fine. |

Snap 5.1: AT command running on serial monitor



**5.1.2 Integration Testing**

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in [software testing](https://en.wikipedia.org/wiki/Software_testing) in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the [compliance](https://en.wikipedia.org/wiki/Regulatory_compliance) of a system or component with specified [functional requirements](https://en.wikipedia.org/wiki/Functional_requirement).[[1]](https://en.wikipedia.org/wiki/Integration_testing#cite_note-1) It occurs after [unit testing](https://en.wikipedia.org/wiki/Unit_testing) and before [validation testing](https://en.wikipedia.org/wiki/Software_verification_and_validation). Integration testing takes as its input [modules](https://en.wikipedia.org/wiki/Module_(programming)) that have been unit tested, groups them in larger aggregates, applies tests defined in an integration [test plan](https://en.wikipedia.org/wiki/Test_plan) to those aggregates, and delivers as its output the integrated system ready for [system testing](https://en.wikipedia.org/wiki/System_testing).

Some different types of integration testing are big-bang, mixed (sandwich), risky-hardest, [top-down, and bottom-up](https://en.wikipedia.org/wiki/Top-down_and_bottom-up_design). Other Integration Patterns[[3]](https://en.wikipedia.org/wiki/Integration_testing#cite_note-3) are: collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

Top-down testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module.

Sandwich testing is an approach to combine top down testing with bottom up testing.

One limitation to this sort of testing is that any conditions not stated in specified integration tests, outside of the confirmation of the execution of design items, will generally not be tested.

**Steps to perform integration testing:**

Step 1: Create a Test Plan.   
Step 2: Create Test Cases and Test Data.   
Step 3: Once the components have been integrated execute the test cases.   
Step 4: Fix the bugs if any and re test the code.  
Step 5: Repeat the test cycle until the components have been successfully integrated.

Table 9.2: Test cases for integration testing.

|  |  |
| --- | --- |
| **Name of the Test** | **Integration testing** |
| Test plan | To check whether the system works properly when all the modules are integrated. |
| Test Data | Pressure sensor value. |

**5.1.3 System testing**

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified [requirements](https://en.wikipedia.org/wiki/Requirements).

System testing takes, as its input, all of the integrated components that have passed [integration testing](https://en.wikipedia.org/wiki/Integration_testing). The purpose of integration testing is to detect any inconsistencies between the units that are integrated together. System testing seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. The actual result is the behavior produced or observed when a component or system is tested.

System testing is performed on the entire system in the context of either [functional requirement](https://en.wikipedia.org/wiki/Functional_requirements) specifications (FRS) or [system requirement](https://en.wikipedia.org/wiki/Requirements_analysis) specification (SRS), or both. System testing tests not only the design, but also the behaviour and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software or hardware requirements specification(s).

Table 5.3: Test cases for Input-Output

|  |  |
| --- | --- |
| **Name of the Test** | **System Testing** |
| Item being tested | All the sensors and wi-fi module connected to Arduino board. |
| Sample Input | Output of the pressure sensor when we apply pressure on it. |
| Expected Output | Arduino board read input and send it to the cloud through wi-fi module. A graph will be plotted according to the input. |
| Actual Output | Application reacts to user inputs in expected manner. |
| Remarks | Successful |

Snap 5.2: Graph Plot for both the pressure sensor.

**CHAPTER 7**

**CONCLUSION AND FUTURE ENHANCEMENTS**

The lack of good quality studies, variation in definition of success and limited follow-up of patients means the success rate of clubfoot treatment using the Ponseti method in sub-Saharan Africa is uncertain. There is need for an agreed definition of good outcome following both the correction and the bracing phase to monitor and evaluate service delivery and identify reasons for poor outcome. It is very important that children who complete the correction phase are followed through the bracing phase and results on success, recurrence and loss to follow up are reported. Studies are also required to document the correlation between clinical outcome, functional outcome and patient/family reported satisfaction

IT and control systems manufacturers are seizing the opportunity of having new novel hardware devices as the “Internet of Things” begins to scale up. As the number of devices continues to increase, more automation will be required for both the consumer (e.g. home and car) and industrial environments. As automation increases in IoT control systems, software and hardware vulnerabilities will also increase. In the near term, data from IoT hardware sensors and devices will be handled by proxy network servers (such as a cellphone) since current end devices and wearables have little or no built-in security. The security of that proxy device will be critical if sensor information needs to be safeguarded. The number of sensors per proxy will eventually become large enough so that it will be inconvenient for users to manage using one separate app per sensor. This implies single appls with control many “things,” creating a data management (and vendor collaboration) problem that may be difficult to resolve. An exponentially larger volume of software will be needed to support the future IoT. The average number of software bugs per line of code has not changed, which means there will also be an exponentially larger volume of exploitable bugs for adversaries. Until there are better standards for privacy protection of personal information and better security guidelines on communication methods and data/cloud storage, security of wearable and other mobility devices will remain poor. More work needs to be spent on designing IoT devices before too many devices are built with default (little or no) security. Physical security will change as well. As self-healing materials and 3D printers gain use in industry, supplychain attacks could introduce malicious effects, especially if new materials and parts are not inspected or tested before use. The main benefits of autonomous capabilities in the future IoT is to extend and complement human performance. Robotic manufacturing and medical nanobots may be useful; however, devices (including robots) run software created by human. The danger of the increased vulnerabilities is not being addressed by security workers at the same rate that vendors are devoting time to innovation. Consider how one might perform security monitoring of thousands of medical nanobots in a human body. The ability to create secure IoT devices and services depends upon the definition of security standards and agreements between vendors. ISPs and telecommunication companies will control access to sensor data “in the cloud” and they cannot provide 100% protection against unauthorized access. IoT user data will be at risk. Diversity of the hardware and software in the future IoT provides strong market competition, but this diversity is also a security issue in that there is no single security architect overseeing the entire “system” of the IoT. The “mission” of the entire IoT “system” was not pre-defined; it is dynamically defined by the demand of the consumer and the response of vendors. Little or no governance exists and current standards are weak. Cooperation and collaboration between vendors is essential for a secure future IoT, and there is no guarantee of success.

**FUTURE OUTCOME**

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Make the device readily available for New Born Baby who has suffered from clubfoot Dieses.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**REFERENCES**

[1] Hosalkar HS, Spiegel DA, Davidson RS (2007) Talipes Equinovarus(Clubfoot). Nelson Textbook of Pediatrics 18: 2777-2778.

[2] M. B. Dobbs and C. A. Gurnett, "Update on clubfoot: etiology and treatment," Clinical Orthopedics & Related Research, Vol. 467, pp. 1146-53, May 2009

[3] J. A. Morcuende, "Congenital idiopathic clubfoot: prevention of late deformity and disability by conservative treatment with the Ponseti technique," Pediatric Annals, Vol. 35, pp. 128-30, 132-6, Feb 2006..

[4] B. D. Roye, J. Hyman, and D. P. Roye, Jr, "Congenital Idiopathic Talipes Equinovarus," Pediatrics in Review, Vol. 25, pp. 124-130, April 1, 2004

[5] I. V. Ponseti and J. Campos, "Observations on pathogenesis and treatment of congenital clubfoot," Clinical Orthopedics & Related Research, Vol. 84, pp. 50-60, May 1972

[6] M. B. S. Brewster, M. Gupta, G. T. R. Pattison, and I. D. Dunn-van der Ploeg, "Ponseti casting: A New Soft Option," Journal of Bone & Joint Surgery British, Vol. 90-B, pp. 1512-1515, November 1, 2008.

[7] D. E. Pittner, K. E. Klingele, and A. C. Beebe, "Treatment of clubfoot with the Ponseti method: a comparison of casting materials," Journal of Pediatric Orthopedics, Vol. 28, pp. 250-3, Mar 2008

[8] A. T. Berman and B. G. Parks, "A comparison of the mechanical properties of fiberglass cast materials and their clinical relevance," Journal of Orthopedic Trauma, Vol. 4, pp. 85-92, 1990

[9] W. M. Mihalko, A. J. Beaudoin, and W. R. Krause, "Mechanical Properties and Material Characteristics of Orthopedic Casting Material," Journal of Orthopedic Trauma, Vol. 3, pp. 57-63, 1989.

[10] B. K. Ng, T. P. Lam, and J. C. Cheng, "Treatment of severe clubfoot with manipulation using synthetic cast material and a foam-casting platform: a preliminary report," Journal of Pediatric Orthopedics B, Vol. 19, pp. 164-70, Mar 2010.

[11] M. G. Zmurko, S. M. Belkoff, and J. E. Herzenberg, "Mechanical evaluation of a soft cast material," Orthopedics, Vol. 20, pp. 693-8, Aug 1997.