

**A SMART-WEARABLE FOR CLINICAL CHARACTERISATION OF MOTOR
SYMPTOMS IN PARKINSON'S DISEASE**

A Project report submitted in partial fulfilment of the requirements for

the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

L. Durga Priyanka (319126512157)

P.Renuka(319126512168)

J.Ashish (319126512150)

G. Vineela(319126512147)

Under the guidance of

Dr. G. Prasanna

Asst. Professor



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(UGC AUTONOMOUS)

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)

Sangivalasa, Bheemili mandal, Visakhapatnam dist. (A.P)

2022-2023

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(UGC AUTONOMOUS)

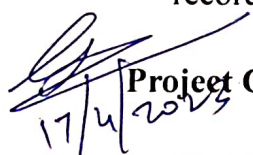
(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)

Sangivalasa, Bheemili mandal, Visakhapatnam dist. (A.P)



CERTIFICATE

This is to certify that the project report entitled "A SMART-WEARABLE FOR CLINICAL CHARACTERISATION OF MOTOR SYMPTOMS IN PARKINSON'S DISEASE" submitted by L.Durga Priyanka(319126512157), P.Renuka(319126512168), J.Ashish(319126512150), G.Vineela(319126512147) in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering of Anil Neerukonda Institute of technology and Sciences(A), Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.


Project Guide

Dr. G. PRASANNA

Asst. Professor

Department of ECE

ANITS

Assistant Professor

Department of E.C.E.

Anil Neerukonda

Institute of Technology & Sciences

Sangivalasa, Visakhapatnam-531 162


Head of the Department

Dr. B. JAGADEESH

Professor & HOD

Department of ECE

ANITS

Head of the Department

Department of E C E

Anil Neerukonda Institute of Technology & Science

Sangivalasa - 531 162

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **Dr. G. Prasanna**, Asst. Professor Department of Electronics and Communication Engineering, ANITS, for his/her guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Dr. B. Jagadeesh**, Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa**, for their encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of ECE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank **all non-teaching staff** of the Department of ECE, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last, but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS

L.DURGA PRIYANKA(319126512157)

P. RENUKA (319126512168)

J.ASHISH (319126512150)

G.VINEELA (319126512147)

CONTENTS

ABSTRACT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Parkinson's Disease

1.1.1 Symptoms of Parkinson's disease	1
1.1.2 Causes of PD	2
1.1.3 Risk factors	3
1.1.4 Statistics on Parkinson's disease	3
1.1.5 Importance of diagnosis of Parkinson's disease	7

1.2 Tremor in human anatomy

1.2.1 Tremor	7
1.2.2 Who is more likely to get tremor	7
1.2.3 Causes for tremors	8
1.2.4 Classification of tremor	8
1.2.5 Categories of tremor	9
1.2.6 Diagnosing tremor	10

1.3 Existing System For Tremor Analysis

1.3.1 Electromyography	10
1.3.2 Dopamine Transporter Scan	11

CHAPTER 2: LITERATURE SURVEY

2.1 Literature survey	12
-----------------------------	----

2.2 Motivation	16
----------------------	----

CHAPTER 3: HARDWARE DESCRIPTION

3.1 PSoC: Programmable System on Chip

3.1.1 Types of PSoC	18
3.1.2 Hardware description of PSoC 3 Kit	18
3.1.3 Hardware description of PSoC 5	20

3.2 ADXL-335 Accelerometer

3.2.1 Feature and benefits of ADXL-335	25
3.2.2 Functional Diagram Of ADXL-335 Sensor	26
3.2.3 PSoC-ADXL Interface	27
3.3 HC-05 Module	
3.3.1 Features	28
3.3.2 Usage	29
3.3.3 PSoc-HC-05 Interface	29
3.4 Power Handler	
3.4.1 Power Bank Circuit	30
3.4.2 Advantages of Power Bank Circuit	31
3.4.3 Li-ion Battery	31
CHAPTER 4: BUILD-IN SOFTWARE	
4.1 PSoC IDE	
4.1.1 PSoC Flow chart	34
4.2 Testing GUI Based Matlab-Interface	
4.2.1 Matlab	35
4.2.2 Matlab Program-Flow chart	36
4.3 MIT App Inventor	
4.3.1 Flow chart	37
4.3.2 Designer	38
4.3.3 Buliding Blocks	40
CHAPTER 5: RESULTS	44
CONCLUSION	47
APPENDIX	
1 PSoC	48

2 Matlab	60
REFERENCES	66

ABSTRACT

Parkinson's disease(PD), a neurodegenerative disorder that manifests both in terms of motor and non motor symptoms and has a need to overcome the limitations of subjective methods. Although research groups worldwide have attempted to design objective assessment for motor symptoms in PD, they have several limitations like observation in only a single assessment regime of PD progression, lower power efficiency and lack of ruggedness. The proposed design is based on state-of-the-art MEMS accelerometer sensors used in conjunction with an advanced mixed signal microcontroller platform-the Infineon® PSoC®. Reconfigurable analog, digital and digital communication components are available on a single chip in the PSoC platform. Custom designing a smart-wearable would find a large user base of rehabilitation specialists and clinicians involved in the diagnosis and treatment of Parkinson's disease.

Key Words : Parkinson's Disease , PSoC , Accelerometer Sensors

LIST OF FIGURES

Figure 1: Prevalence in Household Population	4
Figure 2: Prevalence in Institutional Population.....	4
Figure 3: Percentage of Interactions	5
Figure 4: Expense in a regural house hold	6
Figure 5: PSoC Board	19
Figure 6: Architectural overview of PSoC	21
Figure7: ADXL-335 Sensor	25
Figure 8: Functional diagram of ADXL-335 accelerometer sensor	26
Figure 9: Pin Configuration of ADXL-335 sensor	27
Figure 10: HC-05 Bluetooth Module	28
Figure 11: Circuit Diagram of Power Handler	30
Figure 12: Li-ion Battery	32
Figure 13: Design of Power Bank	33
Figure 14: Design Home Page	38
Figure 15: Page after Interface	39
Figure 16: Building Blocks	40
Figure 17: Insertion of Bluetooth module	44
Figure 18: PSoC board & Power backup tool	45
Figure 19: Power module with sensor and bluetooth module mounted on PSoC module.....	45
Figure 20: Matlab output.....	46

LIST OF TABLES

Table 1: Parameters of USBUART_ Start command	57
---	----

CHAPTER 1

INTRODUCTION

1.1 Parkinson's Disease

Parkinson's disease is a chronic condition that affects both the neurological system and the bodily components that are under the control of the nervous system. The first sign may be a scarcely perceptible tremor in just one hand as the symptoms develop gradually. Although tremors are common, the illness may also cause stiffness or make you move more slowly. This brain disorder causes unintentional or involuntary movements, such as shaking, stiffness, and problems with balance and coordination. Usually, symptoms are minor to begin with and worsen over time. As the condition develops, individuals may have trouble speaking and moving about. Parkinson's disease (PD) is a condition affecting the neurological system that gets worse over time. As nerve cells (neurons) in specific regions of the brain degenerate, get harmed, or pass away, people may have mobility problems, tremor, stiffness in the limbs or trunk of the body, or unstable balance. Parkinson's disease is a progressive neurological ailment that affects the region of the brain responsible for controlling movement. Despite the fact that the brain needs a particular amount of dopamine to control movement, weaker neurons release less dopamine than healthy neurons. Because Parkinson's affects mobility and causes tremors, sluggishness, and trouble walking, it is referred to as a movement condition. Non-movement symptoms are quite common and frequently more debilitating than the ones you can see, such as trouble sleeping, melancholy, and speech problems.

Although there is no known cure for Parkinson's disease, medications may significantly lessen your symptoms. Rarely, your doctor may suggest surgery to treat your symptoms and control particular brain regions.

1.1.1 Symptoms of Parkinson's disease

The indications and symptoms of Parkinson's disease vary from person to person. Early symptoms could be modest and unnoticed. Symptoms commonly begin on one side of the body and are typically severe there even after they begin to affect the limbs on both sides.

- **Tremor:** A tremor is a rhythmic trembling that typically starts in a limb, frequently your hand or fingers. You could wiggle your thumb and forefinger. The term "pill-rolling tremor" describes this. Even when at rest, your hand could shake. While working on a task, the shaking might lessen.
- **Slowed Movements:** Parkinson's disease may cause you to move more slowly over time, making routine chores challenging and time-consuming. When you walk, your steps can

get smaller. It could be challenging to get up from a chair. As you attempt to walk, you can shuffle or drag your feet.

- **Muscle stiffness:** Muscle stiffness can happen anywhere on your body. Your range of motion may be restricted and made painful by the stiff muscles.
- **Impaired posture and balance:** You can develop a hunched posture. Or Parkinson's disease may cause you to trip or have balance issues.
- **Loss of instinctive movements:** You could find it harder to make unintentional movements like blinking, grinning, or swinging your arms while you walk.
- **Speech alterations:** You might speak rapidly, softly, slur, or hesitate before you speak. Your speech may lack the customary speech patterns and seem more monotonous.
- **Changes in writing:** You could find it difficult to write and that your writing is now smaller.

1.1.2 Causes of PD

Parkinson's disease causes some nerve cells (neurons) to eventually degenerate or pass away. Numerous symptoms are brought on by a reduction in the number of neurons in your brain that make the chemical messenger dopamine. As a result of aberrant brain activity brought on by dopamine insufficiency, movement impairment and other Parkinson's disease symptoms get worse.

Although there is no known cause for Parkinson's disease, a number of factors, including:

- **Genes:** Scientists have identified particular genetic mutations that can lead to Parkinson's disease. However, unless multiple family members also have Parkinson's disease, they are unlikely.

Although some gene mutations do appear to increase the likelihood of the ailment, each of these genetic markers has a relatively low risk of Parkinson's disease.

- **Environmental triggers:** There is a slight chance that later-onset Parkinson's disease will be brought on by exposure to specific poisons or environmental factors.

Researchers have also found that individuals with Parkinson's disease undergo a number of changes in their brains, while it is unknown why these changes occur. These alterations consist of:

- The presence of Lewy bodies, which are tiny clusters of specific chemicals found inside brain cells and are a sign of Parkinson's disease. Lewy bodies are what they are, and researchers believe they provide a critical understanding of the mechanisms underlying Parkinson's disease.
- Lewy bodies include a number of substances, but one that is thought to be important is the alpha-synuclein, a naturally occurring and widely dispersed protein. (a-synuclein). It is clumped and found in all Lewy bodies; cells cannot break it down. Researchers looking into Parkinson's illness are currently focusing a lot on this.

1.1.3 Risk factors

- **Age:** Young adults are rarely afflicted by Parkinson's disease. It typically appears in middle or later life, and as you age, your risk of developing it increases. Usually, persons with the condition are 60 years of age or older. Young people with Parkinson's disease may benefit from genetic counselling to help them with family planning options. Work, social situations, and medicine side effects are additional characteristics that are distinct from those of an older person with Parkinson's disease and call for special treatment.
- **Heredity:** Having a close relative with Parkinson's disease increases your risk of developing it yourself. Your risks are still quite low unless you have a sizable number of family members who are affected with Parkinson's disease.
- **Sex:** Men are more frequently affected by Parkinson's disease than women are.
- **Toxic exposure:** Regular exposure to pesticides and herbicides may marginally raise your risk of developing Parkinson's disease.

1.1.4 Statistics on Parkinson's disease

Parkinson's disease is the most common neurological disorder after Alzheimer's disease. Nearly 90,000 people are diagnosed with PD each year in the United States. PD affects more than 10 million people globally. Every year, Parkinson's disease (PD) is diagnosed in roughly 60,000 Americans. Parkinson's disease becomes more prevalent as people get older, yet estimates show that only 4% of cases are discovered prior to the age of 50. Current estimates indicate that PD contributed to 329,000 fatalities in 2019—an increase of more than 100% since 2000—and 5.8 million years of disability-adjusted life—an increase of 81% since 2000.

Figure 1: Prevalence in Household Population

Parkinson's disease prevalence in households, by sexe and age group, for people aged 45 and older.

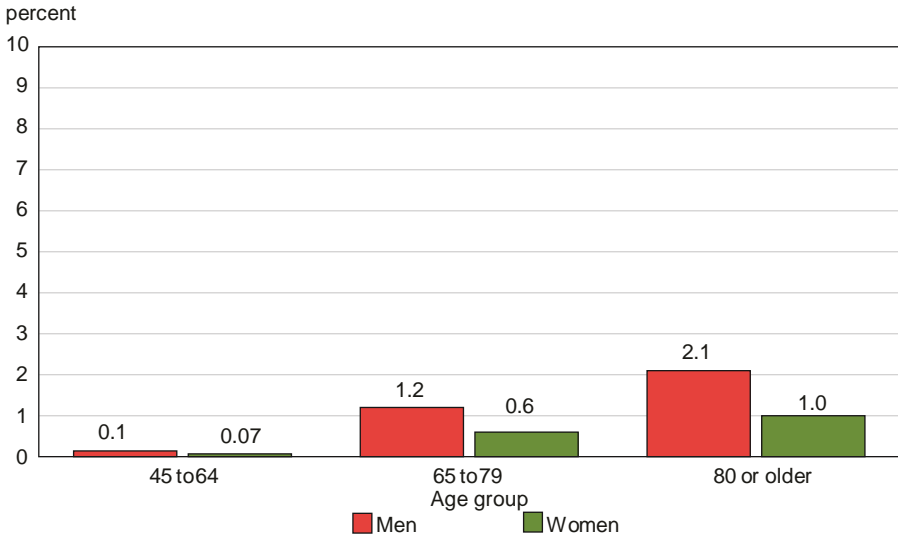
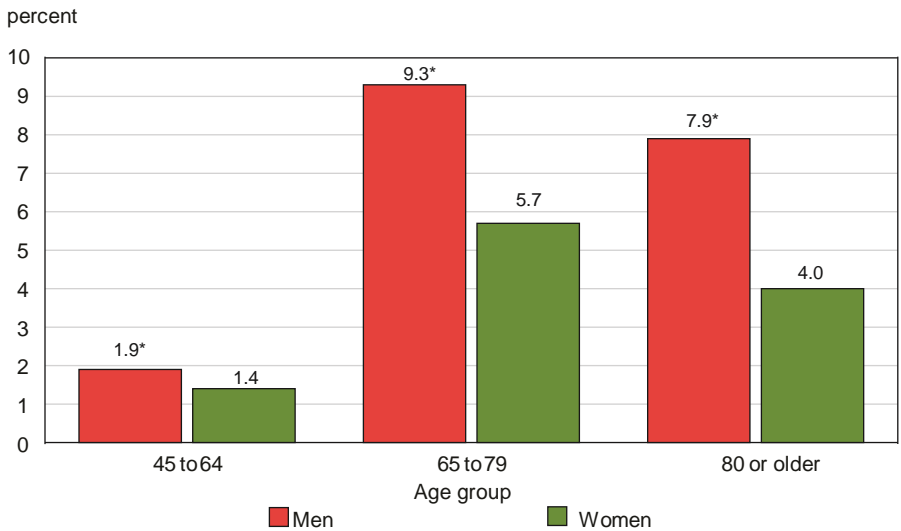


Figure 2: Prevalence in Institutional Population

Parkinson's disease prevalence in those living in institutions, broken down by sex and by age groups, for those who are at least 45.



Source: Statistics Canada, CANSIM Table 105-1305.

Parkinson's disease frequency A diagnosis of Parkinson's disease was made in about 55,000 Canadians who were 18 or older and lived in private homes. Similar to the CCHS prevalence estimate from 2000–2001, this constituted 0.2% (95% CI: 0.2%, 0.3%) of the household population.⁶ Additionally, 4.9% (95% CI: 4.8%, 5.0%) of the 12,500 inhabitants at long-term residential care institutions reported having Parkinson's disease. In households, 79% of those with Parkinson's were 65 or older; in institutions, 97% of those with the condition were in the same age bracket. Overall, men had a higher prevalence of Parkinson's disease than women: 0.3% versus 0.2% (p 0.05) for people living in private homes, and 6.6% versus 4.0% (p 0.05) for people living in institutions. Parkinson's disease prevalence generally increased with age,

however in the community of institutionalised people, it reduced in the oldest age group. (Figures 1 and 2). This may be due to older individuals in institutions having more severe diseases and dying more frequently than those living in private homes.⁷ In addition, Parkinson's patients have been found to die earlier when they are male and have symptoms such as severe motor impairment, psychosis, and dementia.

Awareness in people

Communication with others Numerous persons with Parkinson's reported feeling humiliated (43%) or excluded (29%) as a result of their illness. Others seemed to shun them (19%) or feel uneasy about them (28%) for some people. Over half (58%) said they had at least one of these worries. (Figure 3)

Figure 3: Percentage of Interactions

Percentage of households with adults 18 and older who have Parkinson's disease and whose relationships with others are impacted by the condition, Canada excluding territories, 201

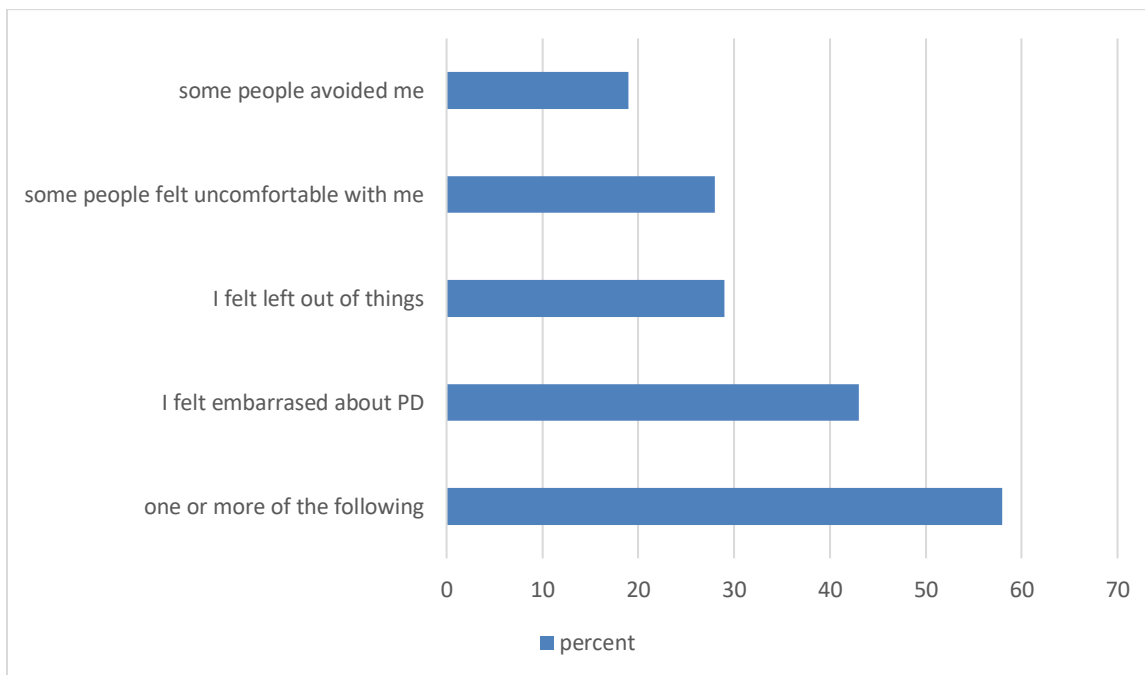
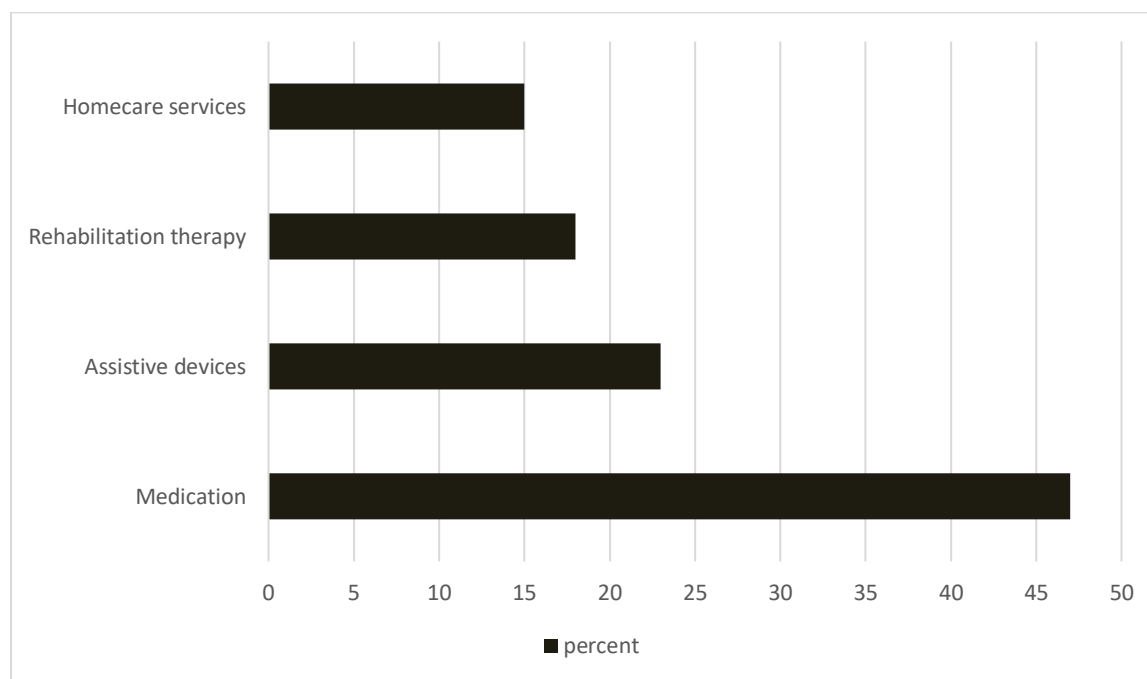


Figure 4: Expense in a regular house hold



percent whose expenses were \$500 or more

Survey

75 Parkinson's disease (PD) patients in a movement disorders programme participated in a survey. We questioned patients and carers about the two most troublesome PD-related issues, coping strategies for these issues, the motor and non-motor PD-related issues that patients needed the most support with, and what should be included in a comprehensive assistance programme for PD patients and carers. To summarise the responses, we employed qualitative data analysis approaches.

Results: Tremors, limited mobility, pain, imbalance, loss of energy/fatigue, having to give up previously enjoyed activities, dysarthria, and anxiety or depression were reported as the most upsetting issues. Medication, exercise, instrumental or practical support, and emotional support were frequently mentioned methods of coping with various difficulties. Respondents most frequently cited melancholy and anxiety, "nothing," or cognitive issues when questioned directly about which non-motor disorders aroused the greatest need for aid. Participants recommended including education, physical activity, and emotional support in a comprehensive assistance programme for people with Parkinson's disease and those who care for them.

Conclusions: The survey's results show the variety of PD experiences that patients can have and the significance of coping mechanisms for both motor and non-motor symptoms.

1.1.5 Importance of Parkinson's disease diagnosis

Effective Parkinson's disease treatment and symptom management depend on an early and precise diagnosis. Parkinson's disease is often determined by a professional, such as a neurologist, after reviewing the patient's signs and symptoms, reviewing their medical history, and performing a neurological and physical examination. Despite the lack of a specific test for Parkinson's, a timely diagnosis enables medical professionals to make a more precise diagnosis and begin treatment sooner.

As the disease worsens over time and the symptoms increase, a proper diagnosis is crucial. Early identification may aid in halting the disease's progression and enabling appropriate symptom management with medication, exercise, and other treatments. Early diagnosis can also assist patients and their families with coping strategies and care and support needs planning.

1.2 Tremors in human anatomy

1.2.1 Tremor

Involuntary, rhythmic movements of a bodily component are called tremors. They most frequently affect the hands or arms, but they can also affect the head, neck, vocal chords, chest, and legs. Numerous conditions, including neurological abnormalities, adverse drug reactions, stress, anxiety, and exhaustion, can result in tremors. Tremors may occasionally signal a more severe underlying medical condition, such as Parkinson's disease, multiple sclerosis, or a stroke.

Resting tremors, postural tremors, intention tremors, and task-specific tremors are only a few of the several types of tremors. As the injured body part is at rest, resting tremors start to occur. They usually go away as the person starts to move. When someone is keeping a position that defies gravity, such as holding their arm outstretched, postural tremors might occur. When a person is trying to move a body part, such as reaching for something, they experience intention tremors. Task-specific tremors happen while performing particular tasks, including writing or playing an instrument.

Daily tasks might be disrupted by tremors, which in some situations can also result in social isolation and despair. Medication, therapy, and lifestyle modifications such as abstaining from coffee and getting enough sleep are all possible treatments for tremors. Surgery may be required in some circumstances to alleviate symptoms.

1.2.2 Who is more likely to get tremor?

Tremor can affect anyone at any age, however it most frequently affects middle-aged and older persons. Men and women are typically equally affected by the condition.

Tremor typically results from an issue in the deep brain regions that regulate movement. Although some tremor types seem to be inherited and run in families, the majority of tremor types have no known origin.

Tremor can happen on its own or be a sign of several neurological conditions. among the recognised causes are:

- Medicines: Amphetamines, corticosteroids, several treatments for psychiatric and neurological diseases, as well as some asthma medications, can all cause tremors.
- Abuse or withdrawal from alcohol: This may result in tremors and an overactive nervous system.
- Illnesses or illnesses including Parkinson's disease, multiple sclerosis, traumatic brain injury, or stroke

1.2.3 Causes for tremors

Another neurological condition called tremor produces shaking in one or more body parts, most frequently the hands. According to the National Institute of Neurological Disorders and Stroke, unplanned (involuntary) muscular contractions, which might happen at different times with gaps in between them, are what generate the rhythmic pattern.

The most typical kind of tremors, known as essential tremors, usually start off gradually and are more noticeable on one side of the body. They typically start in the hands, affecting one or both hands, and get worse with movement. Emotional stress, exhaustion, coffee, and severe temperatures can all exacerbate essential tremors.

Tremors can be caused by a number of risk factors. For instance, a mutated gene that can be passed down from just one parent may be the source of essential tremor. Medical diseases include Parkinson's disease, stroke, multiple sclerosis, traumatic brain injury, an overactive thyroid, and mercury poisoning from food or the environment can all induce other types of tremors. Additionally, overstimulating the nervous system with excessive caffeine or some drugs might exacerbate the reasons of tremors.

1.2.4 Classifications (types) of tremor

Tremors fall into one of two categories: resting or active.

When the muscle is relaxed, as when your hands are on your lap, a resting tremor takes place. Even at rest, your hands, arms, or legs can tremble. The tremor frequently only impacts the hand or fingers. Those who have Parkinson's disease frequently experience this kind of tremor.

Action tremor happens when a muscle contracts voluntarily. Action tremors are the most prevalent types of tremors. Action tremors can be divided into a number of categories, many of which overlap.

- Postural tremor happens when you maintain a stance that defies gravity, such as putting your arms out in front of you
- Any intentional movement, such as moving your wrists up and down or closing and opening your eyes, is related with kinetic tremor.
- When you make an intentional movement towards a target, like moving your finger to touch your nose, intention tremor begins.
- Task-specific tremor only manifests during highly skilled, purposeful activities like speaking or scribbling.
- An isometric tremor happens when a voluntary muscle contraction takes place without any accompanying movement, such as when holding a heavy book still.

1.2.5 Categories of tremor

The two factors that most distinguish tremor types are appearance and origin. There are more than 20 different types of tremors, with the following being some of the most prevalent:

Fundamental tremor One of the most prevalent movement diseases is essential tremor, often known as benign essential tremor or familial tremor in the past. Its defining characteristic is a tremor in the hands and arms both when moving and when rest. It might also have an impact on your head, voice, and gait. Even though the tremor can arise at any age, it typically first manifests in adolescent or middle age. (between ages 40 and 50). Either it will gradually get worse or it will remain moderate.

Cerebellar tremor: A cerebellar tremor is a gradual, obvious shaking of the hands, feet, arms, or legs that happens after an intentional motion, such as pressing a button. It is brought on by harm to the cerebellum and the circuits that connect it to other parts of the brain, commonly from a stroke or tumor, injury from a disease or an inherited condition, or chronic alcoholism-related damage.

Functional tremor, also known as psychogenic tremor, can take on any tremor-like appearance. Its symptoms can vary, but they frequently start and cease abruptly and can affect any portion of the body. When you are under stress, the tremor gets worse; when you are distracted, it gets better or goes away.

Parkinsonian tremor: Although not all persons with Parkinson's disease exhibit tremor, Parkinsonian tremor is a prevalent and one of the early indications of the disease. When the hands are relaxed and shaking is most evident, it may appear as though someone is trying to roll a pill between the thumb and a finger. Also susceptible to Parkinson's tremor are the chin, lips, face, and legs. The tremor may at first only affect one limb or one side of the body, but as the illness worsens, it could affect both sides as well. Stress or intense emotions might amplify the tremor.

1.2.6 Tremor Diagnosing

Your doctor will examine you physically and go through your health history. Your doctor might also recommend a neurological examination, which would assess your speech, balance, reflexes, muscle tone, and strength:

- The location of the tremor on the body; • Whether the tremor happens while the muscles are at rest or in motion. (and if it occurs on one or both sides of the body)
- The tremor's appearance (tremor frequency and visibility)

Additionally, in order to rule out specific medications or other potential causes of your tremor, you could be requested to provide blood or urine samples. If there is brain injury present, diagnostic imaging may assist identify it. A muscle or nerve issue may be found via an electromyogram, which analyses involuntary muscle activity and muscle response to nerve stimulation.

To identify any functional limitations, such as issues with handwriting or the capacity to hold a fork or cup, additional tests may be used. You might be required to complete a series of exercises or tasks, including placing your finger on their nose's tip or drawing a spiral.

1.3 Existing Systems for Hand Tremor Analysis

1.3.1 Electromyography

Electromyography (EMG) is a diagnostic test that is used to assess the state of the muscles and the nerve cells that control them, the motor neurons. Results from an EMG may suggest issues with nerve-to-muscle signal transmission, muscular dysfunction, or tremor-causing nerves. Motor neurons send electrical signals that cause muscles to contract. These impulses are transformed by an EMG into graphs, noises, or numerical values, which are then assessed by a professional utilising minuscule devices known as electrodes. A needle EMG involves inserting a needle electrode into a muscle to capture the electrical activity of that muscle. Using electrode stickers affixed to the skin (surface electrodes), the nerve conduction study, a part of an EMG, evaluates the speed and strength of impulses travelling between two or more sites.

EMG outcomes are:

- Muscle disorders like polymyositis or muscular dystrophy Nerve-muscle connection-disturbing diseases like myasthenia gravis
- Peripheral nerve illnesses, such as carpal tunnel syndrome or peripheral neuropathies, affect the nerves outside the spinal cord.

- Conditions that damage the motor neurons in the brain or spinal cord, such as polio or Amyotrophic Lateral Sclerosis
- Nerve root conditions, such as a herniated spinal disc.

RISKS

EMG is a procedure with a low risk of complications. The risk of bleeding, infection, and nerve injury is slight if a needle electrode is inserted. During a needle electrode examination of the muscles along the chest wall, there is a very small possibility that air could seep into the area between the lungs and chest wall, resulting in the collapse of a lung (pneumothorax).

1.3.2 DOPAMINE TRANSPORTER SCAN (Da T Scan)

The term "DaT Scan" (also known as a "Dopamine Transporter Scan") is frequently used to describe a diagnostic procedure to check for dopaminergic neuron loss in the striatum. The brand name of Ioflupane(123I), which is utilised in the study, may also be referred to by this title. The radiopharmaceutical Ioflupane(123I), which binds to dopamine transporters, forms the foundation of the scan principle. (DaT). Single-photon emission computed tomography (SPECT), which employs specialised gamma-cameras to build a pictorial picture of the distribution of dopamine transporters in the brain, is then used to detect the signal from them. When we are unsure of the cause of a patient's tremor, DaTSCAN is recommended. This approach can distinguish between Parkinson's symptoms and essential tremor, but it cannot determine whether the issue is Parkinson's disease or Multiple System Atrophy.

A patient should take two iodine tablets at first and wait an hour. These medications are crucial because they stop radioactive chemicals from building up in the thyroid gland. The patient must then wait for four hours after receiving an injection of radiopharmaceutical into his shoulder after waiting for an hour. As the substance's concentration rises, a gamma camera that is positioned around his head scans it. The entire evaluation is non-invasive and lasts roughly 30-45 minutes. If a patient takes any of the medications listed below, they must be stopped a few days or weeks before to the DaTSCAN, but only after consulting their physician. Patients don't need to spend the night in the hospital because the examination only takes a few hours, but they do need to drink a lot more than usual and use the loo more frequently. It is crucial for the quick removal of radioactive materials from the body.

CHAPTER 2

Literature Survey and Motivation

2.1 Literature Survey

1. Millions of people worldwide are afflicted by PD, a widespread and incapacitating condition that is characterized by both motor and non-motor symptoms. In order to monitor and control the pathology, sensor devices that are inexpensive, low-power, discrete, and accurate in their measurements are used for disease detection. A review made by Erika Rovini [1] has five key areas of interest: early diagnosis, tremor, body motion analysis, motor fluctuations (ON-OFF phases), and home and long-term monitoring. It focuses on wearable devices for PD applications. Wearable sensors are necessary for clinicians to perform early diagnosis, differential diagnosis, and objective measurement of symptoms across time. The use of inertial sensors like accelerometers (ACC) and gyroscopes (GYRO), combined with advancements in short-range communication technologies (i.e., Bluetooth, Zigbee), is now practical and meets the needs of people with chronic disorders because it offers low power consumption, unobtrusiveness, light weight, and ease of use. This review's objective is to provide readers with a thorough grasp of the benefits and drawbacks of wearable sensor technologies for PD applications. The use of gloves into which the sensors are inserted does not seem to be the ideal solution in terms of the wearability of the devices because of the problems related to the noises emitted.

2. The objective examination of symptoms can help neurologists make accurate assessments, enhancing the standard of treatment for patients. There is a study reviewed by Filippo Cavallo[4] that sought to determine whether kinematic features may be used to forecast PD severity levels by creating data-driven models based on regression methods. This presented a suggestion for additional aspects of highly effective wearable devices. Thirteen motor task performances were observed based on the clinical evaluation, and 179 kinematic features were derived from the inertial motor data. This predictive model may serve as a decision-support tool for an accurate, unbiased evaluation of the severity of Parkinson's disease (PD) based on mobility performance, enhancing patient monitoring over time. This research is a necessary first step towards a data-driven objective assessment of the pathology that could help the neurologist increase the precision of the PD assessments. Undoubtedly, a data-driven assessment can aid in lowering the inter-rater variability that frequently influences the diagnosis of Parkinson's disease, leading to more accurate assessments, and possibly improving the standard of treatment for PD patients. The sensor system can only be used for subjective analysis; precise values cannot be acquired.

3. There is a study that reviewed by N. Kostikis[7] that suggest a useful smart phone-based technique to precisely measure upper limb tremor in people with Parkinson's disease (PD). This suggested technique uses a mobile smart phone platform to measure PD-induced hand tremor. This study aims to look at using a smart phone-based method to measure PD-induced hand tremor. This method involves measuring PD hand tremor using the accelerometer and gyroscope sensors built with the phone. This technique for quantifying upper limb Parkinsonian tremor can be utilized at home by patients to track their own development as well as in a clinical

environment to aid the doctor's job by providing her with an accurate assessment tool. This tool is inexpensive, platform independent, non-intrusive, and easy to use. With the aid of signal processing, machine learning, and motion tracking technology, PD patients can have their diagnosis and overall quality of life evaluated as well as any motor anomalies found. Additionally, this is only employed for early detection, necessitating the invention of a diagnostic tool for other tremor applications.

4. Since tremor is the most prevalent motor condition associated with Parkinson's disease (PD), detecting it is essential for managing and treating PD patients. The current method of diagnosis introduced by George Rigas[10] is depending on subject-specific clinical assessment, which has trouble identifying tiny tremor symptoms. Here, a collection of accelerometers put on various patient body parts is suggested as an automated way for assessing both resting and action/postural tremor. Based on features used from the collected signals and hidden Markov models, tremor kind (resting/action postural) and severity are estimated. The accelerometer of each sensor records a signal corresponding to a certain axis. (x, y, and z). This technique may accurately identify the presence of the characteristic Parkinsonian tremor, which may aid in the early identification of the condition. A strategy like this can also be useful for determining how the disease is progressing. As a result, this technique may serve as the foundation for consistent daily and long-term monitoring of PD patient tremor activity. The findings include: tremor severity can be quantified with 87% accuracy; tremor can be distinguished from other Parkinson's disease motor symptoms during daily activities; and tremor can be distinguished from postural tremor when one is resting. For larger sample sets that are utilized to verify the gait metrics, a few theories are put forth. This makes the difference in PD performance easy to distinguish.

5. Simple tests like timed up and go (TUG) or walking trials can be used to assess locomotion in order to assess treatment and make an early diagnosis of Parkinson's disease in patients. (PD). Either complicated motion laboratory settings or straightforward timing outputs using stop watches are often employed techniques in clinics. An innovative technology is proposed by Benoit Mariani[14] based on on-shoe wearable sensors and a processing system, TUG and gait tests provide result measurements characterizing PD motor symptoms. In order to distinguish between control and PD participants, it is necessary to demonstrate and evaluate the usage of on-shoe wearable sensors and a specialized algorithm that can measure both TUG and long-distance walking. Along with objectively quantifying 3-D gait parameters, this approach also offers outcome measurements that enable the detection of gait beginning, steady state, turning, and termination. Comparing PD patients in ON and OFF states with control subjects demonstrates the method's potential. Following the experiment, the mean and standard deviation were used to estimate the accuracy and precision of the retrieved spatiotemporal parameters. (STD) . In the control and ON groups, turning was found to be the longer phase during TUG, but not in the OFF group. In comparison to the literature, the TUG durations attained in the OFF group were slightly shorter in the ON group. Common motor function tests for PD were instrumented using

on-shoe wearable sensors. This study offers fresh insights into the viability of home movement disorder patient monitoring utilizing wearable technology. To establish the significance of the novel gait metrics, additional clinical research and statistical analysis with larger sample sizes are necessary.

6. A small wireless attitude and heading reference system has been designed in order to build a body sensor network for the quantitative monitoring of Parkinson's disease patients' motor function during rehabilitation sessions. These technologies are extremely appealing for mobile health applications since they enable patients to complete rehabilitation exercises at home while also receiving feedback on their performance. The paper proposed by Michele Caldara [17] discusses the single node's performance, the wearable network's idiosyncrasies, and the special software created just for the Extended Timed-Up-and-Go test. This demonstrates how the development of Attitude and Heading Reference Systems (AHRS), made possible by the combination of sensors and a microcontroller capable of determining the 3D orientation, is driven by the availability of tiny, low-cost, and high precision MEMS sensors. Such devices are particularly appealing as body motion tracking systems since they offer a wireless interface. The parameters can be used for software analysis to monitor the effects of pharmacological therapy or to evaluate the patient's motor function during the rehabilitation phase. The device can monitor a large number of factors during exercise, including asymmetries in gait, posture, tremors, and total and intermediate exercise execution times, according to preliminary findings from the ongoing clinical research. The detection of freezing of gait (FOG), which is difficult to predict, has led to the implementation of experimental methods for all freezing whereas only a few studies have used TUG for FOG detection that does not include specific activities.

7. A crippling sign of Parkinson's disease (PD), freezing of gait (FOG), a paroxysmal loco motor block, is linked to a higher risk of falling and early nursing home placement. Unbiased raters count the number of FOG episodes from video as part of their clinical assessment of FOG in Parkinson's disease (PD). The current approach was developed by Tiffany R. Morris[19] and compared the results of clinical evaluation of FOG using computer-generated cartoons during a timed up and go test with the "gold standard" of clinical video assessment using a group of ten trained raters from four PD clinics. This finding suggests that in addition to the number of FOG episodes, percent time frozen should also be considered in order to more properly convey the severity of FOG. This study compared the usefulness of clinical FOG assessment utilising these 3D computer cartoons to the existing "gold standard" of clinical FOG assessment from footage of the actual patient. In order to compare objective measures of FOG with clinical observation during routine tasks, the goal of this study was to verify a technique for clinical assessment of FOG in circumstances where video or direct observation is not feasible. The results present a potential method for confirming accelerometry-based FOG diagnosis outside of the clinic and demonstrate that evaluating FOG clinically using computer-generated animations derived from lower-body acceleration data is viable. In addition to being easier to deploy, video communicates a lot more information than lower-limb animation, from upper body action to the patient's and

attending clinician's facial expressions. Our computer-generated animation method has disadvantages in terms of complexity and the potential for artifactual motion.

8. Another sign of PD is a loss of postural control. Accelerometers can be a viable and trustworthy alternative to force plates, which are the traditional way of posture study. These two measurement methods are both tremor-sensitive. Tremor impacts postural measurements, which could result in inaccurate findings or conclusions. For the removal of tremors, linear low-pass filters (LPFs) are frequently used. An alternate method was proposed by Sabato Mellone[22] based on Hilbert-Huang transformation (HHT). This technique uses high-pass filtering to remove high-frequency noise and the tremor component from accelerometer records of trunk sway during calm stance. Our first method for signal preprocessing used the low pass Butterworth filter, one of the most frequently used in this type of applications due to its maximally flat frequency response in the pass band. HHT is characterised by an algorithm that can be divided into two steps: 1) a preprocessing step called EMD that acts as a dyadic filter bank to decompose the original signal; and 2) a second step where the decomposition is put through the Hilbert transform to create an amplitude-frequency-time distribution. The resultant time-frequency distribution of the amplitudes is the Hilbert spectrum of the signal. As the illness worsens, the frequency of tremors tends to decrease as the amplitude increases. This indicates that when the disease severity rises, the adoption of an effective tremor control technique becomes increasingly important. Based on the findings mentioned in this research, it is advised to use HHT-based filtering as an effective and reliable tremor removal tool since it preserves local dynamics without reducing frequency bandwidth. Future research will look at the proposed framework from the opposite perspective with the intention of using the pure tremor component for tremor analysis, which is a useful tool. This element is derived from acceleration of trunk sway.

9. By expanding the functionality of physiological monitoring equipment, advancements in mobile and electronic healthcare are revolutionising the engagement of both doctors and patients in the contemporary healthcare system. Despite notable development in the monitoring device sector, there has been no general use of this technology in clinical settings. The review introduced by Connolly E S[25] main purpose is to provide a summary of the advancements and clinical applications of wearable smart sensors. It examined the research on real-time home tracking devices, connected gadgets, sensors, trackers, tele-monitoring, wireless technology, and their use by physicians. Smart wearable sensors are trustworthy and helpful for preventative measures in various fields of medicine, including cardiopulmonary, vascular, endocrine, neurological function, and rehabilitation therapy. The accuracy and utility of these sensors have also been demonstrated in the fields of rehabilitation medicine and preoperative monitoring. These sensors are simple to maintain and are growing in accuracy and dependability for patient care. In certain trials, patients were even subjected to numerous sensors to aggregate data and get the best results. The accuracy was raised to 87% by combining the information from accelerometers attached to the wrist and hip. They also discovered that two accelerometers, positioned on the chest and hip, are all that are necessary to achieve comparable findings. Other

chest-worn accelerometers can identify snoring and breathing characteristics to diagnose sleep apnea.

10. The development of new technology for assessing Parkinson's disease (PD) outcomes has opened the door to the prospect of measurement taking place in patients' homes while they are living freely and going about their daily lives as usual. This systematic review, which Lynn Rochester[28] created, intends to provide a broad overview of the technology being used to investigate the effects of participant activities during free-living in a home environment on Parkinson's disease (PD). The utilisation of technology, the establishment of a home or homelike environment, the measurement of any motor or non-motor aspect of PD relevant to daily living activities, and unrestricted/unscripted participant activities are all covered. Adults with PD diagnosis are also included. Among the variables the technology examined were gait, tremor, physical activity, bradykinesia, dyskinesia, and motor fluctuations. Wearable technology, specifically inertial measurement units, was used to do this. Additionally, it examined typing, falling, sleep, and daily life activities. Many organisations utilise different approaches to conduct home-based, free-living testing in PD, with an emphasis particularly on motor symptoms and sleep. The most recent information about PD free-living home-based sensor testing is summarised here. This review is interested in the information gathered from unplanned behaviours in free living, whether those behaviours are ongoing or not. As far as we are aware, this systematic review is the first to evaluate the body of knowledge around the use of wearable and non-wearable technology that passively monitors unstructured behaviours (free-living) in the home or home-like environment in order to develop outcome measures in PD. There are positive activities seeking to advance this field in a collaborative and sensible approach (with industry, academia, and others) towards the implementation of health technology to monitor results in PD.

2.2 Motivation

Smart wearable sensor systems are most widely used tools to detect the body postures and movements and used to diagnose it. However based on many studies many limitations are being identified and some of them are ; many of the systems are high power consuming tools, wireless communication design is yet to be designed, memory installation option is highly required to store the results, low sampling rate one of the biggest issue as it is delaying the detection and hence medication problems are being faced. So, we are trying to overcome this limitations by developing a tool which is a portable device and having a power backup system. We are also trying to make it storage capable and by using sensors we are also trying to increase the sampling in order to not delay the diagnosis.

CHAPTER 3

HARDWARE DESCRIPTION

3.1 PsoC: Programmable System on Chip

3.1.1 Types of PSoC:

There are several types of PSoC chips available, including:

1. PSoC 1: The first generation of PSoC devices, featuring a 8-bit microcontroller core, and a large parts of analog and digital parts.
2. PSoC 3: The second generation of PSoC devices, featuring a 32-bit ARM Cortex-M3 core, and more advanced analog and digital components.
3. PSoC 4: A low-power, cost-effective version of PSoC, featuring a 32-bit ARM Cortex-M0 core, and a range of analog and digital components.
4. PSoC 5: A high-performance PSoC device, featuring a 32-bit ARM Cortex-M3 core, and a wide variety of advanced analog and digital components.
5. PSoC 6: A dual-core PSoC device, featuring a 32-bit ARM Cortex-M4 and Cortex-M0+ cores, and a range of advanced analog and digital components.

Each of these types of PSoC devices has its own unique features and capabilities, and is designed for different applications and use cases.

3.1.2 Hardware description of PsoC3 kit:

The PSoC 3 kit is a development platform for Cypress's PSoC 3 family of microcontrollers. It includes an evaluation board, USB cable, and a CD with software and documentation. Here is a brief hardware description:

1. PSoC 3 microcontroller: The heart of the kit is the PSoC 3 microcontroller, which combines a powerful 8-bit CPU, analog and digital peripherals, and programmable logic in a single chip.
2. Evaluation board: The evaluation board includes headers for connecting external peripherals and a variety of built-in features, including:

- USB 2.0 interface
- Voltage regulators
- Crystal oscillator
- User LEDs and switches
- Potentiometer for analog input
- Prototyping area for custom circuits

3. USB cable: The kit comes with a USB cable for programming and communication between the PSoC 3 and a host computer.

4. CD: The kit includes a CD with Cypress's PSoC Creator software, documentation, and sample projects. PSoC Creator is an integrated development environment that allows you to design and program PSoC 3 projects using graphical drag-and-drop components and C code.

Overall, the PSoC 3 kit is a versatile and powerful platform for developing embedded systems

Figure5 PSoC 3 Board



Features:

- PSoC 3 microcontroller: The kit includes a PSoC 3 microcontroller, which is a powerful and flexible device that integrates programmable analog and digital peripherals onto a single chip.
- Integrated programming and debugging: The PSoC 3 kit includes an integrated programmer and debugger, which allows you to program and debug the PSoC 3 device using the USB interface.
- USB interface: The kit includes a USB interface that can be used to communicate with the PSoC 3 device and to power the board.
- Expansion headers: The kit includes headers that allow you to connect external hardware, such as sensors or actuators, to the PSoC 3 device.

- On-board peripherals: The kit includes several on-board peripherals, including buttons, LEDs, a potentiometer, and an accelerometer, which can be used for testing and development.
- Software development tools: The PSoC 3 kit is supported by Cypress's PSoC Creator software development environment, which includes a graphical user interface for designing and configuring the PSoC 3 device, as well as a C compiler and debugging tools.

Voltage Regulator:

- The PSoC3 is a microcontroller from Cypress Semiconductor that includes a voltage regulator circuit on the chip. The voltage regulator is responsible for maintaining a stable voltage level for the PSoC3's internal circuitry and any external devices connected to it.
- The PSoC3 voltage regulator is a linear regulator, which means it controls voltage using a series pass transistor. The voltage regulator circuit includes a feedback mechanism that monitors the output voltage and modifies the pass transistor to maintain a constant output voltage.
- The voltage regulator circuit has an input voltage, typically ranging from 2.5V to 5.5V, and an output voltage that is regulated to a fixed value, such as 3.3V or 5V. In most cases, the feedback mechanism compares the output voltage to a reference value using an op-amp and produces an error signal that is used to modify the pass transistor.
- The voltage regulator circuit also includes protection features such as overvoltage protection, over current protection, and thermal protection to prevent damage to the chip and any external devices connected to it.

3.1.3 Hardware description of PSoC5:

The PSoC 5LP (Programmable System-on-Chip 5LP) is a family of microcontroller units (MCUs) developed by Cypress Semiconductor. The PSoC 5LP kit is a development kit based on this MCU and is designed to facilitate rapid prototyping and development of embedded systems.

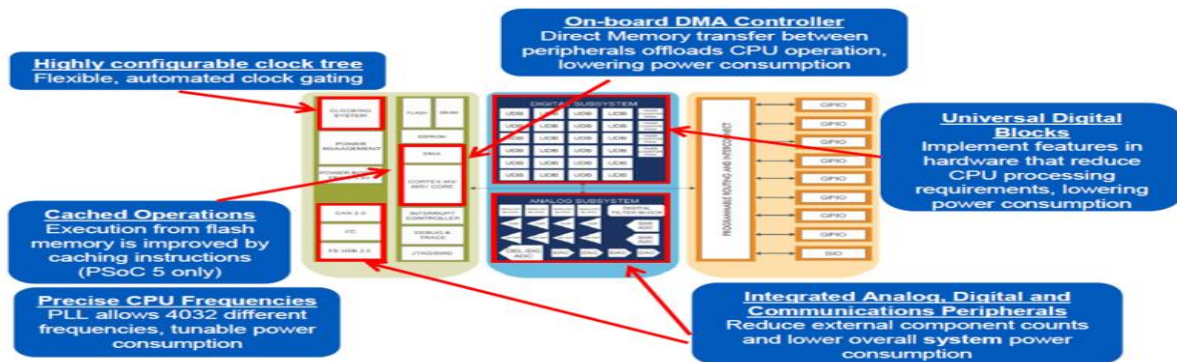
Here are some hardware specifications of the PSoC 5LP kit:

- CPU: PSoC 5LP (ARM Cortex-M3 core, 80 MHz clock speed)
- Memory: 256 KB of flash memory, 64 KB of SRAM
- Connectivity: USB 2.0 Full Speed (12 Mbps), I2C, SPI, UART, CAN, LIN, and USB-UART bridge
- Analog inputs: 20-bit SAR ADC (up to 1 Msps), programmable gain amplifier (PGA), and 12-bit DAC

- Digital inputs/outputs: Up to 64 GPIO pins (programmable as digital inputs/outputs or analog inputs)
- Other features: Programmable logic (PLD) for custom digital circuit design, CapSense capacitive touch sensing technology, and support for external memory through a serial interface (SPI or I2C)

The PSoC 5LP kit also comes with a variety of on-board sensors and peripherals, including an accelerometer, a temperature sensor, and an ambient light sensor. Additionally, the kit includes an on-board debugger and programmer, allowing for easy programming and debugging of the PSoC 5LP MCU.

Figure 6: Architectural overview of PSoC



<https://slideplayer.com/slide/4468774/14/images/9/Designed+for+Low+Power%2FLow+Voltage.jpg>

ARM Cortex-M3 core:

The PSoC5 microcontroller from Cypress Semiconductor is one application that heavily utilizes the ARM Cortex-M3 32-bit microcontroller core. The ARM Cortex-M3 core is integrated with programmable analogue and digital blocks in the programmable system-on-chip known as the PSoC5, which enables designers to construct unique microcontroller solutions that are catered to their unique requirements.

The ARM Cortex-M3 core used in the PSoC5 provides high performance and low power consumption, making it well-suited for a wide range of applications. It includes features such as a 3-stage pipeline, hardware single-cycle multiplication and division, and a nested vectored interrupt controller, which help to optimize performance while minimizing power consumption.

The PSoC5 also includes a variety of configurable analog and digital blocks that can be used to implement a wide range of functionality, such as ADCs, DACs, op-amps, comparators, timers,

counters, PWMs, and more. These blocks can be configured using Cypress's graphical design tool, allowing designers to easily create custom microcontroller solutions that meet their specific requirements.

Overall, the ARM Cortex-M3 core used in the PSoC5 provides a powerful and flexible platform for building custom microcontroller solutions for a wide range of applications.

1.USB 2.0:

USB 2.0 Full Speed is a standard communication speed for USB devices, which provides a maximum data transfer rate of 12 Mbps. This speed is suitable for many applications, such as connecting printers, scanners, and other peripherals to a computer.

In PSoC5, the USB interface is implemented using a USB controller, which handles the low-level USB communication protocols. The USB controller in PSoC5 supports USB 2.0 Full Speed communication and is compliant with the USB 2.0 specification.

To use the USB interface in PSoC5, you need to configure the USB controller and implement the necessary firmware to handle USB data transfer. Cypress provides a USB Component in its PSoC Creator software, which simplifies the process of configuring and implementing the USB interface in PSoC5. With this component, you can quickly create USB applications, including HID, CDC, and MSC, by configuring the USB controller and implementing the firmware for your application.

2.I2C:

I2C, or Inter-Integrated Circuit, is a popular communication protocol used to transfer data between integrated circuits in a system. PSoC5 is a family of microcontrollers developed by Cypress Semiconductor that includes support for the I2C protocol.

To use I2C on a PSoC5 microcontroller, you need to configure the I2C component in your PSoC Creator project. You can do this by adding the I2C component to your project and configuring its parameters such as the slave address, clock frequency, and data rate.

Once the I2C component is configured, you can use the I2C APIs provided by Cypress Semiconductor to implement the I2C communication in your code. These APIs allow you to write and read data to and from the I2C bus using the PSoC5 microcontroller.

3.SPI:

SPI" stands for Serial Peripheral Interface, which is a synchronous serial communication protocol used to communicate with peripherals.

The PSoC 5 supports the SPI interface and includes a hardware module called the Serial Communication Block (SCB), which provides configurable serial communication capabilities, including SPI. The SPI module in PSoC 5 supports both Master and Slave modes of operation.

The SPI interface on the PSoC 5 has the following key features:

- Configurable data width from 8 to 16 bits
- Configurable clock polarity and phase
- Configurable data transfer rate up to 24 Mbps
- Multi-master support
- Interrupt-driven or DMA-based data transfer

To use the SPI module on the PSoC 5, you would typically configure the module's registers to set the desired mode of operation, data transfer rate, clock polarity and phase, and other parameters. Then, you would use the appropriate APIs provided by the PSoC 5 development environment (such as PSoC Creator or Modus Toolbox) to interact with the SPI module and send/receive data to/from the connected peripheral devices.

4.UART:

The PSoC 5 is a microcontroller from Cypress Semiconductor that has a Universal Asynchronous Receiver/Transmitter (UART) module. The UART module in PSoC 5 provides serial communication capability, which is commonly used to communicate with other devices, such as sensors, displays, and other microcontrollers.

To use the UART module in PSoC 5, you will need to configure the module using the PSoC Creator IDE. Here are the steps to configure the UART module in PSoC 5:

1. Open PSoC Creator and create a new project.
2. Drag and drop the UART component from the Component Catalog onto the schematic.
3. Double-click the UART component to open the configuration window.
4. Configure the UART module as per your requirements, such as the baud rate, parity, stop bits, and data width.
5. Once you have configured the UART module, you can use the APIs provided by the UART driver to send and receive data over the serial port.

5.CAN(Controller Area Network): The CAN communication protocol is widely used in automotive and industrial applications, and Cypress's PSoC 5 implementation of CAN complies fully with ISO 11898-1. But CAN might not be present in every PSoC5 device.

6.LIN:

LIN (Local Interconnect Network) is a communication protocol commonly used in automotive and industrial applications for connecting sensors, actuators, and other devices to a central control unit. PSoC5 is a family of microcontrollers from Cypress Semiconductor that supports LIN communication.

To implement LIN communication on PSoC5, you would typically use a LIN transceiver IC connected to the PSoC5 UART (Universal Asynchronous Receiver/Transmitter) interface. The LIN transceiver acts as a physical layer interface between the PSoC5 and the LIN bus, while the PSoC5 firmware handles the higher-level protocol tasks such as message transmission, reception, and error checking.

Cypress provides a LIN component in their PSoC Creator software that simplifies the implementation of LIN communication on PSoC5. The component includes pre-built LIN firmware libraries and configuration tools to easily set up the LIN communication parameters.

Overall, LIN on PSoC5 provides a flexible and cost-effective solution for implementing communication in various automotive and industrial applications.

7.USB-UART bridge:

The USB-UART bridge of PSoC 5 is a hardware component that enables communication between the PSoC 5 microcontroller and a computer via a USB interface. It is essentially a UART (Universal Asynchronous Receiver-Transmitter) interface that is connected to a USB controller.

The USB-UART bridge is integrated into the PSoC 5 architecture and can be configured through software using the PSoC Creator IDE. It allows for serial communication between the microcontroller and the computer, and is commonly used for debugging and data transfer purposes.

To use the USB-UART bridge, you need to connect the PSoC 5 microcontroller to the computer via a USB cable. Then, you can use a terminal program on the computer to send and receive data to and from the microcontroller.

Overall, the USB-UART bridge of PSoC 5 provides a convenient and easy-to-use interface for communication between the microcontroller and a computer, making it a popular choice for many embedded system applications.

3.2 ADXL-335 Accelerometer Sensor:

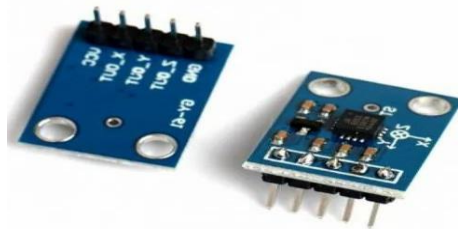
A 3 – axis accelerometer and 3 –axis gyroscope, small, low power, ± 3 g.

- The ADXL335 is a signal-conditioned 3-axis accelerometer that is small, thin, fully featured, and low-power. The instrument measures acceleration across a full-scale range of at least 3 g. It may track both the static acceleration of gravity and the dynamic acceleration caused by motion, shock, or vibration in tilt sensing applications.
- The user can select the bandwidth of the accelerometer using the CX, CY, and CZ capacitors, which are situated at the XOUT, YOUT, and ZOUT pins. Bandwidths can be selected to fit the application with a range of 0.5 Hz to 1600 Hz for the X, Y, and Z axes and 0.5 Hz to 550 Hz for the Z axis.
- A compact, low profile, 16-lead, plastic lead frame chip scale package (LFCSP_LQ) with dimensions of 4 mm by 4 mm by 1.45 mm is available for the ADXL335.

3.2.1 Feature and benefits of ADXL – 335:

- Small, low-profile package: 1.45 mm LFCSP; 4 mm; 4 mm;
- 350 μ A of low power (typical)
- Operation from a single supply, 1.8 V to 3.6 V
- three-axis sensing
- ten thousand g shock survival
- outstanding temperature stability
- A single capacitor per axis for the BW adjustment
- Lead-free and RoHS/WEEE compliant

Figure7 ADXL-335 Sensor



<https://5.imimg.com/data5/FQ/CX/EH/SELLER-4167793/adx1335-module-3-axis-accelerometer-1000x1000.jpg>

3.2.2 FUNCTIONAL DIAGRAM OF ADXL-335 SENSOR

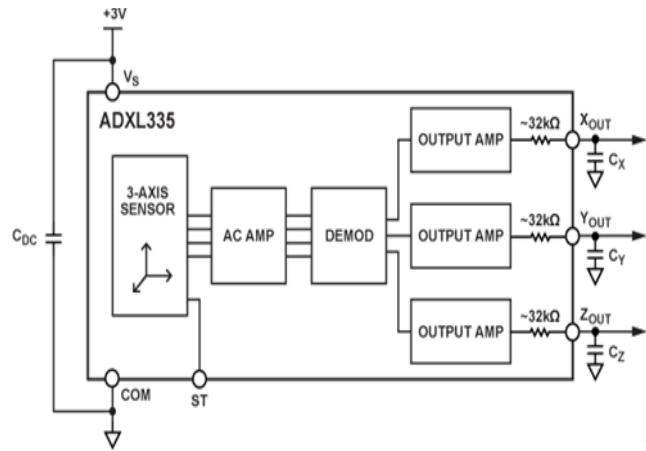


Figure8 : Functional Diagram of ADLX -335 accelerometer sensor

PIN CONFIGURATION AND DESCRIPTION

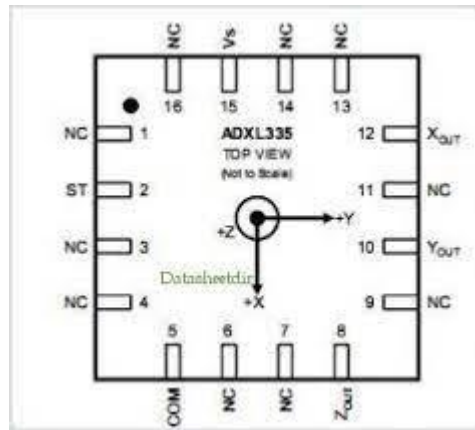


Figure9: Pin configuration of ADXL -335 sensor

3.2.3 PSoC-ADXL Interface:

To interface PsoC with the ADXL sensor, you can follow these general steps:

Connect the ADXL sensor to the PsoC development board. The ADXL sensor typically uses an I2C or SPI interface, so make sure that you connect the appropriate pins to the corresponding pins on the PsoC board.

1. Configure the PsoC development environment to communicate with the ADXL sensor. You will need to set up the appropriate communication protocol (I2C or SPI) and configure the PsoC pins to act as the appropriate communication lines.
2. Write PsoC firmware to read data from the ADXL sensor. Depending on the communication protocol used, you will need to implement the appropriate communication protocol to read data from the ADXL sensor. For example, if using I2C, you will need to implement code to send I2C commands to the ADXL sensor and read data back.
3. Process the data obtained from the ADXL sensor. Once you have successfully read data from the ADXL sensor, you can process the data as needed. This might include filtering, scaling, or converting the data into a usable format for your application.
4. Use the processed data in your PsoC application. Finally, you can use the data obtained from the ADXL sensor in your PsoC application. This might involve using the data to control other components of your system or sending the data to a host computer for further analysis.

3.3 HC-05 Module:

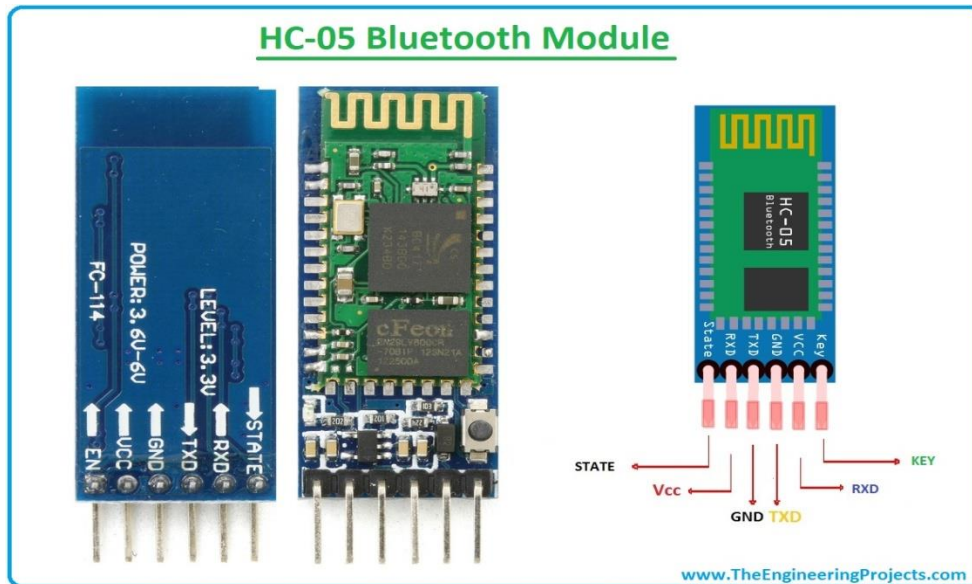
The HC-05 module is a popular Bluetooth module that allows wireless communication between electronic devices such as microcontrollers, smart phones, laptops, and tablets. It uses Bluetooth 2.0 and can be used as a master or slave device.

The HC-05 module typically comes in a small form factor, with a serial interface for easy integration into various electronic projects. It can communicate with other Bluetooth devices using the Serial Port Profile (SPP), which makes it easy to establish a wireless communication link between two devices without needing to worry about the lower-level Bluetooth protocol stack.

The HC-05 module can be configured using AT commands, allowing the user to set parameters such as the device name, baud rate, and PIN code. It can also be set up to work in a variety of modes, including master mode, slave mode, and loopback mode.

Overall, the HC-05 module is a versatile and easy-to-use Bluetooth module that is widely used in hobbyist and professional electronic projects.

Figure 10: HC-05 Bluetooth Module



<https://images.theengineeringprojects.com/image/main/2019/10/HC-05-Bluetooth-Module-Pinout-Datasheet-Features-Applications-1.jpg>

The HC-05 is a Bluetooth module that allows wireless communication between devices. Here are some of its features and potential uses:

3.3.1 Features:

- Uses Bluetooth 2.0 and EDR (Enhanced Data Rate) technology
- Can function as either a master or slave device
- Supports point-to-point and point-to-multipoint connections
- Operates at a frequency range of 2.4GHz ISM band
- Offers a range of up to 10 meters
- Has a built-in antenna
- Can be powered by a 3.6V to 6V DC power source
- Can communicate at a baud rate of up to 1382400bps
- Supports various data formats, including ASCII and binary

3.3.2 Usage:

- Can be used in wireless data transmission projects, such as sending sensor data to a microcontroller or sending commands to a robot
- Can be used for wireless communication between two microcontrollers or between a microcontroller and a computer
- Can be used for wireless communication between a smartphone and a microcontroller or computer
- Can be used for wireless communication in Internet of Things (IoT) projects and in home automation systems

- Overall, the HC-05 Bluetooth module is a versatile and useful tool for wireless communication in a variety of projects.

3.3.3 PSoC - HC-05 Interface:

Interfacing PSoC with the HC-05 module can enable wireless communication between the PSoC and other devices that are Bluetooth-enabled.

To interface PSoC with HC-05 module, you will need to follow these steps:

1. Join the PSoC microcontroller and the HC-05 module. To accomplish this, join the TX pin of the module to the RX pin of the PSoC and the RX pin of the module to the TX pin of the PSoC. Additionally, join the module's VCC pin to the PSoC's power supply pin and the ground pin of the module to the latter's ground pin.
2. Setup the UART module on the PSoC microcontroller. This will make it possible for the microcontroller and HC-05 module to communicate over the serial port. The UART module can be set up and initialization code generated using the PSoC Creator programme.
3. Configure the HC-05 module for communication with the PSoC microcontroller. This can be done by sending AT commands to the module. You can use the PSoC microcontroller to send these commands to the module over the serial interface. Some of the commonly used AT commands for configuring the HC-05 module include setting the Bluetooth device name, setting the baud rate, and enabling or disabling the Bluetooth module.
4. Write code to enable communication between the PSoC microcontroller and the HC-05 module. This can be done by writing code to send and receive data over the UART interface. You can use the PSoC Creator software to generate the code for sending and receiving data over UART.

Once you have completed these steps, you should be able to interface PSoC with HC-05 module and use it for wireless communication between devices.

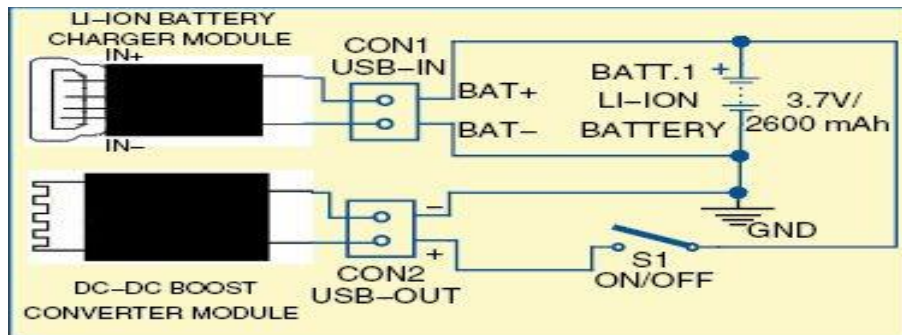
3.4 Power Handler:

In the event of a power outage or disturbance, a power handler, often referred to as a transfer switch, is a device used in power backup systems to transition power supply from the primary power source to a backup power source, such as a generator or uninterruptible power supply (UPS).

The power handler monitors the incoming power supply and automatically switches to the backup power source when it detects a power outage or a drop in voltage or frequency beyond a set threshold. This ensures uninterrupted power supply to critical equipment and systems during a power outage or disruption.

3.4.1 Power Bank Circuit:

Figure11 Circuit Diagram of Power Handler



https://www.electronicsforu.com/wp-content/uploads/2015/11/F54_1.jpg

A power bank is a portable device that is used to store electrical energy in its battery cells and then use that energy to charge other electronic devices such as smart phones, tablets, and laptops. The circuitry inside a power bank is designed to efficiently manage and transfer the energy stored in its battery cells to the electronic devices being charged.

The main components of a power bank circuit include:

1. **Battery:** The battery is the primary source of electrical energy that is stored in the power bank. It can be made up of one or more lithium-ion or lithium-polymer cells depending on the capacity of the power bank.
2. **Charging and Discharging Circuit:** The charging circuit is responsible for charging the battery cells when the power bank is connected to a power source such as a USB port or wall adapter. The discharging circuit is responsible for regulating the output voltage and current to the device being charged.
3. **Protection Circuit:** The protection circuit is designed to prevent overcharging, over-discharging, short circuits, and overheating, which can cause damage to the battery cells or the electronic devices being charged.
4. **Voltage Regulator:** The voltage regulator ensures that the output voltage of the power bank remains constant even when the battery voltage drops.
5. **LED Indicator:** The LED indicator is used to show the remaining battery capacity of the power bank.

Power banks can come in various shapes and sizes, and the circuitry may vary depending on the specific features and functionality of the power bank. However, the basic components outlined above are present in most power bank circuits.

3.4.2 Advantages of Power Bank Circuit:

A power bank circuit is an electronic circuit that is designed to provide portable power for charging mobile devices and other portable electronic devices. Some advantages of using a power bank circuit include:

- **Portability:** A power bank circuit is a compact and portable device that can be easily carried around, making it ideal for use on the go or in remote locations where access to a power source is limited.
- **Convenience:** A power bank circuit provides a convenient way to charge your mobile devices without having to rely on a wall outlet or other external power source.
- **Universal compatibility:** A power bank circuit is designed to work with a wide range of mobile devices, including smart phones, tablets, and other portable electronic devices.
- **High capacity:** A power bank circuit can provide a high capacity of power, allowing you to charge your devices multiple times before needing to be recharged itself.
- **Multiple charging ports:** A power bank circuit can have multiple charging ports, allowing you to charge multiple devices at the same time.
- **Safety features:** A power bank circuit can include safety features such as overcharge protection, short circuit protection, and overvoltage protection, which help to prevent damage to your devices and the power bank itself.
- **Cost-effective:** A power bank circuit is generally more cost-effective than purchasing individual battery packs for each device, especially when considering the high capacity and universal compatibility of the power bank circuit.

3.4.3 Li-ion Battery:

Lithium ions are the main element in the electrolyte of a lithium-ion battery, which is a rechargeable battery. It's a kind of rechargeable battery frequently used in portable devices, electric cars, and other uses where high energy density and lightweight are desired.

Lithium-ion batteries can store a lot of energy despite being relatively light and small in size due to their high energy density. The fact that they can be recharged makes them a popular choice for portable electronics. Lithium-ion batteries are composed of a positive electrode (cathode) and a negative electrode (anode), which are separated by an electrolyte. Lithium ions alternatively flow between the two electrodes during charge-discharge cycles.

Benefits of lithium-ion batteries include their high energy density, low self-discharge rate, and lack of memory effect. In addition, they have a longer lifespan than standard rechargeable batteries. However, they might be prone to thermal runaway and sensitive to high temperatures,

which could cause a fire or an explosion. Use caution when using and storing them to maintain safety.

Figure 12: Li-ion Battery



<https://tse3.mm.bing.net/th?id=OIP.goqMk-WTLEqpfJiypf3vIQHaHa&pid=Api&P=0>

Care must be used when handling the Li-ion battery because overcharging puts the battery at risk of catching fire. The Li-ion battery is charged using specialised ICs like the TP4056 IC. which, as soon as the battery is fully charged, automatically disconnects it from the power source. A single programmable resistor can be used to charge a Li-ion battery with the help of the IC. According to the battery that has to be charged, the charging current can be adjusted by adding a resistor and capacitor. Additionally, the IC has internal heat management and current control.

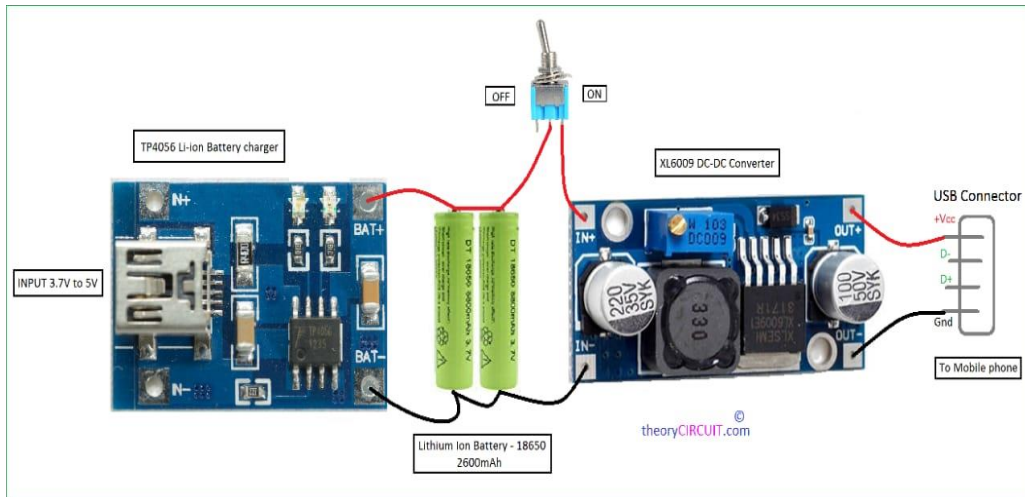


Figure 13: Design of Power Bank

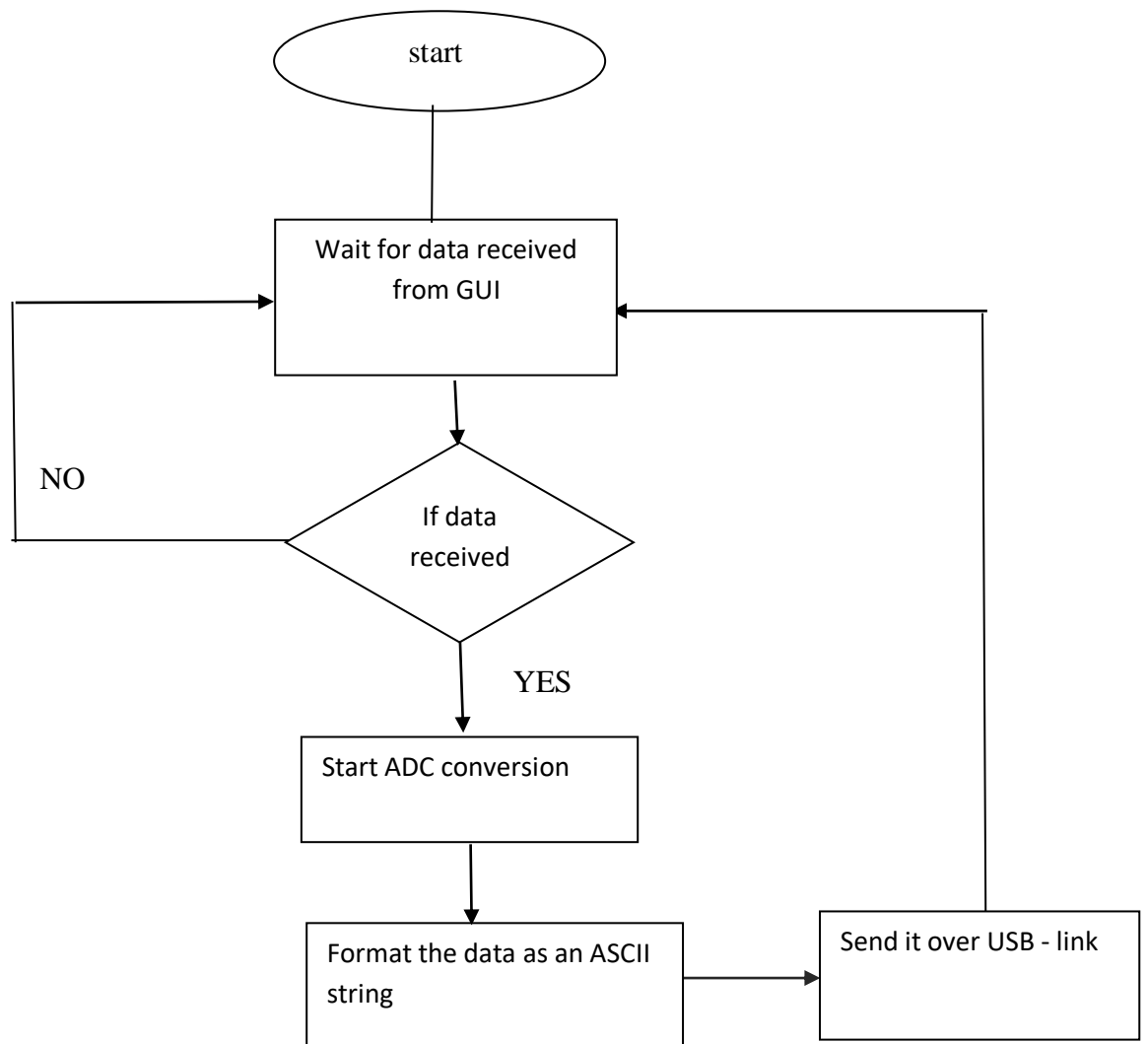
CHAPTER 4

PSoC FIRMWARER DESIGN

4.1 Psoc IDE

A software tool called PSoC IDE (Integrated Development Environment) is used to create programmes for PSoC microcontrollers. Cypress Semiconductor Corporation created the series of microcontrollers known as PSoC, or Programmable System-on-Chip. Uniquely, PSoC microcontrollers may be completely customised to meet the needs of any application by combining programmable logic, configurable analogue and digital blocks, and a powerful CPU core.

4.1.1 Psoc Flow chart



4.2 TESTING GUI BASED MATLAB – INTERFACE

4.2.1 Matlab

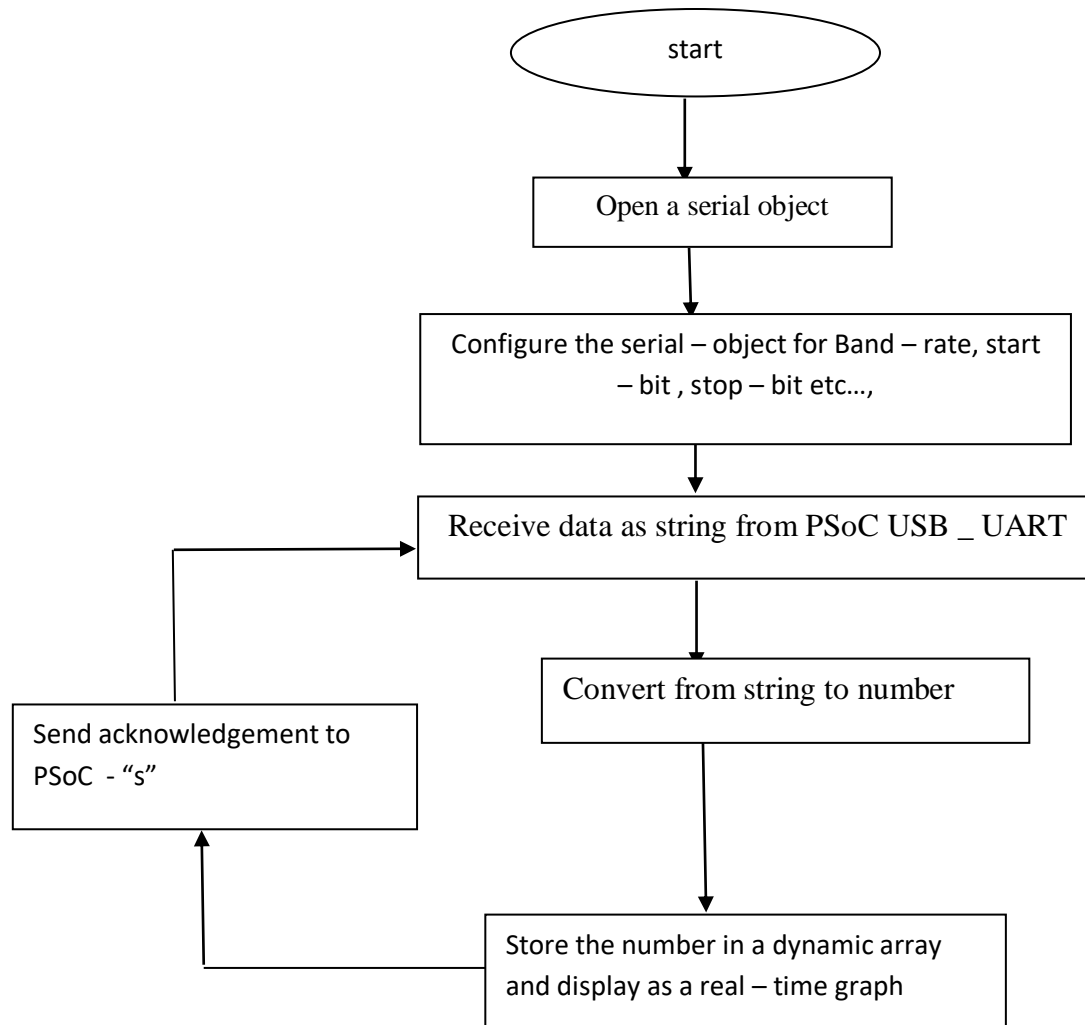
The MathWorks company created the high-level programming language and interactive environment known as MATLAB (short for Matrix Laboratory). It is frequently used for data analysis, visualization, and numerical computation in engineering, science, and mathematics. With built-in libraries for a variety of applications, MATLAB offers a flexible and powerful environment for algorithm development, data analysis, and modelling.

In addition to a graphical user interface (GUI) for creating and editing graphical user interfaces and programs, MATLAB has a command-line interface for entering commands and running scripts. It offers functions for conducting operations on these data types, such as linear algebra, signal processing, and statistics, and supports a broad variety of data types, including matrices, arrays, structures, and cell arrays.

The creation of apps for a range of platforms, such as desktop PCs, online browsers, and mobile devices, is also supported by MATLAB. It has a sizable user base and a vast library of add-ons and toolboxes that add extra capability for certain applications like image processing, control system design, and machine learning.

Overall, MATLAB is an effective tool for data analysis, modeling, and simulation, offering a wide range of built-in functions and libraries, a versatile programming environment, and a vibrant user base. It is widely utilised in industry and academia for a wide range of applications, from engineering design and analysis to scientific research.

4.2.2 MATLAB PROGRAM – FLOW CHART

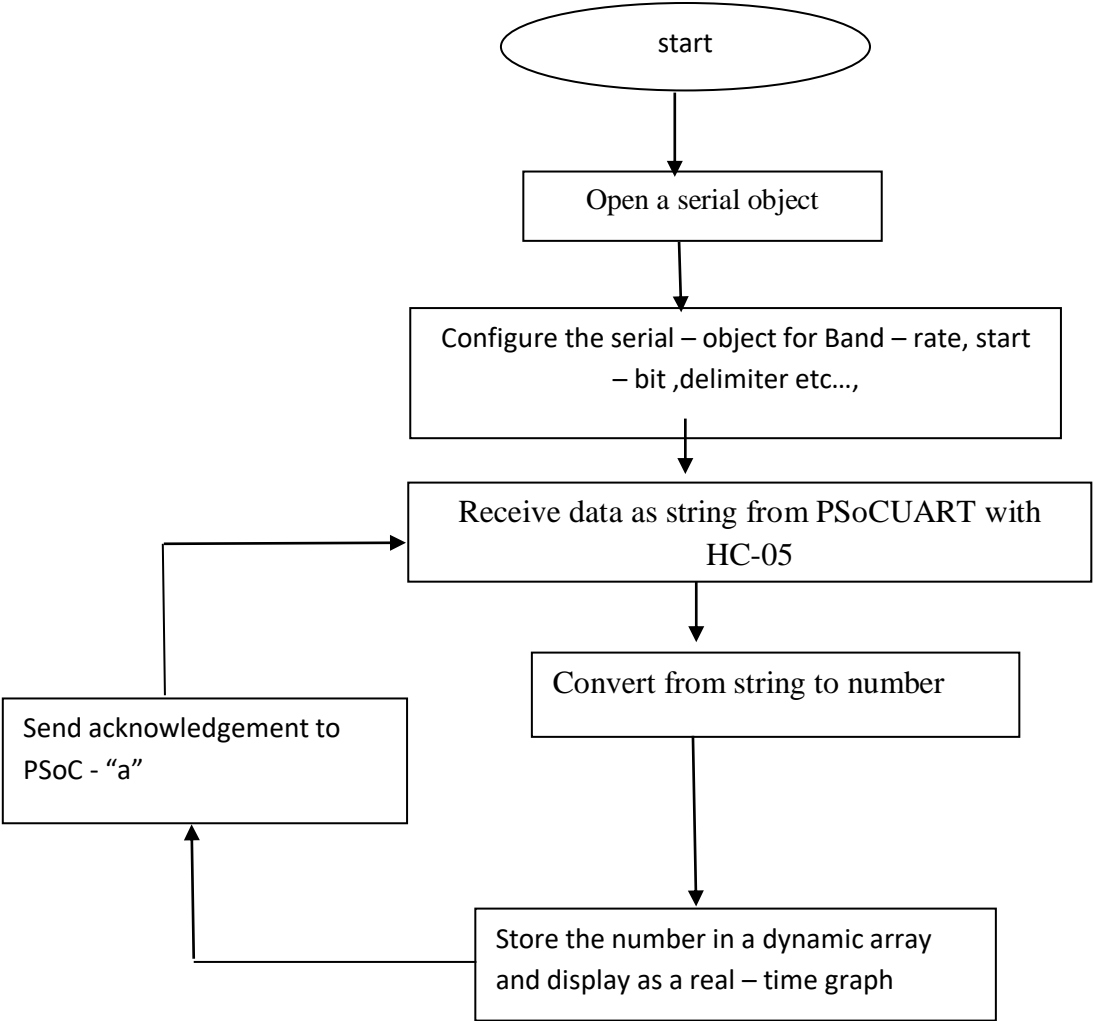


4.3 MIT APPINVENTOR

MIT App Inventor is a visual, blocks-based programming language and development environment for creating mobile applications for Android devices. It was created by the MIT Center for Mobile Learning and is designed to make app development accessible to anyone, even those without prior programming experience

App Inventor provides a web-based development environment where users can create, design, and test their mobile applications without the need for complex programming languages.

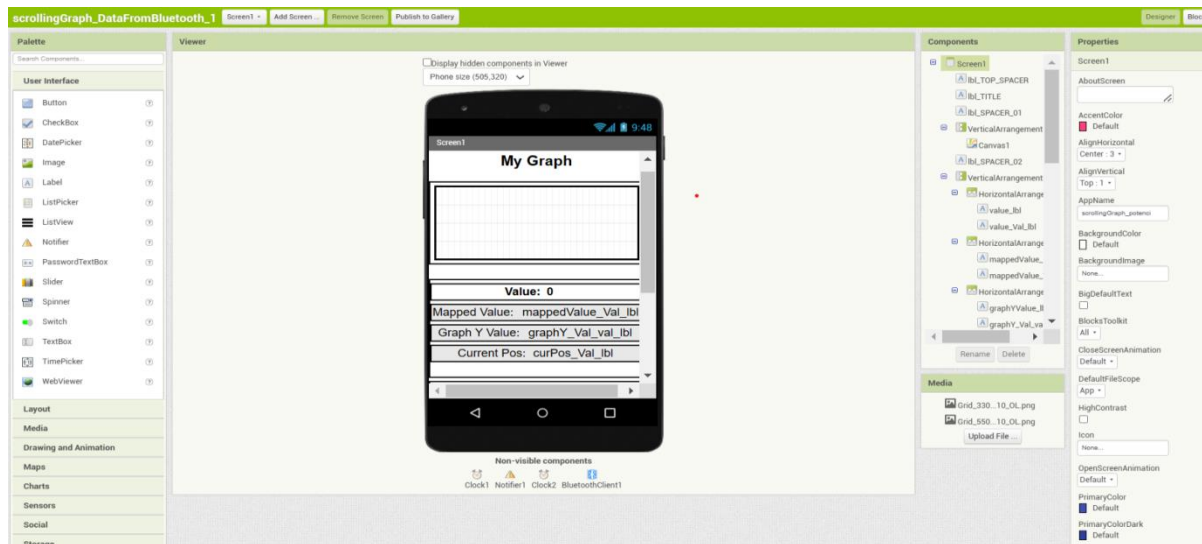
4.3.1 Flow chart



MIT App Inventor has two main components: the Designer and the Blocks Editor.

4.3.2 Design

Figure 14: Design Home Page



Users of MIT App Inventor can create the user interface for their Android mobile app using the Designer visual interface. To design the layout of the app, users can drag and drop different components onto the Designer canvas, including buttons, labels, text boxes, photos, and other UI elements. Users can easily see how their app will appear and work thanks to the Designer's visual user interface representation.

To design a user interface in MIT App Inventor, you can start by selecting the "Designer" tab in the top left corner of the screen. From there, you can drag and drop components onto the screen, such as buttons, text boxes, images, and other user interface elements.

Once you have added your desired components, you can arrange them by dragging and dropping them to the desired location on the screen. You can also adjust their properties to create your desired user interface.

For instance, you can add a button to your user interface and then edit its text, color, size, and other properties. You can also add an image to your interface and set its size, position, and other properties.

MIT App Inventor also allows you to add different types of layouts to your user interface. You can add horizontal and vertical layouts to organize your components in a specific way. You can also add scrollable layouts to enable users to scroll through a long list of items.

To add functionality to your user interface, you can use the blocks editor in MIT App Inventor. The blocks editor lets you create code blocks that define the behavior of your components. For instance, you can create a code block that defines the action to be taken when a button is clicked.

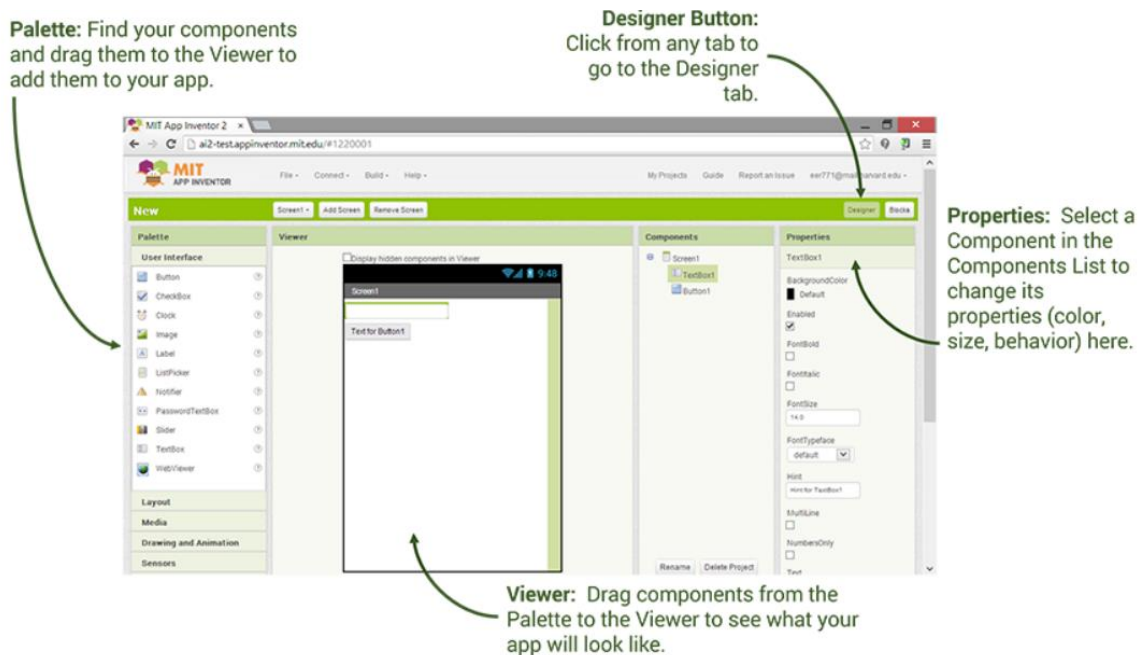
Overall, designing a user interface in MIT App Inventor is a simple process that involves dragging and dropping components, arranging them, and adjusting their properties. With a little creativity and some knowledge of the blocks editor, you can create a user interface that meets your needs and engages your users.

To export your MIT App Inventor user interface to an APK file, you can select the "Export Selected Project (.apk) to my computer" option under the "Projects" menu. This will generate an APK file that you can install on your Android device or distribute to others.

Before exporting your project, make sure that you have tested it thoroughly and that all components and functionality are working as expected. You should also check that your user interface is responsive and looks good on different screen sizes and orientations.

It is also a good idea to add a version number and release notes to your APK file to keep track of changes and improvements over time. Once you have exported your project, you can share it with others or publish it to the Google Play Store for others to download and use.

Figure 15: Page after Interface



4.3.3 Building blocks

Figure 16

```
to updateGraph
do
  set global comment to "A random value is used in this example."
  set global value to random integer from get global minValue to get global maxValue
  set value Val lbl to get initial value
  set global mappedValue to call m Returns a random integer between the upper bound and the lower bound. The bounds will be clipped to be than 2**30.
  outSTART
  outEND
  get global canvasHeight
  set mappedValue Val lbl to get global mappedValue
  set global mappedValue to get global canvasHeight * get global mappedValue
  set graphY Val val lbl to get global mappedValue
  if
    get global Graph_Current X_Pos <= get global Graph_maxPos
  then
    set global Graph_Current X_Pos to get global Graph_Current X_Pos + 1
    replace list item list get global Graph_Y_Val_List
    index get global Graph_Current X_Pos
    replacement get global mappedValue
  else
    for each i from 1
    to get global Graph_maxPos - 1
    by 1
    do
      replace list item list get global Graph_Y_Val_List
      index get i
      replacement select list item list get global Graph_Y_Val_List
      index get i + 1
    replace list item list get global Graph_Y_Val_List
    index get global Graph_maxPos
    replacement get global mappedValue
  set global comment to "redrawing the canvas like this, even when not sc..."
  call Canvas1 Clear
  for each i from 1
  to get global Graph_Current X_Pos - 1
  by 1
  do
    call Canvas1 DrawLine
    x1 get i * get global Graph_Resolution + get global Graph_Resolution
    y1 select list item list get global Graph_Y_Val_List
    index get i
    x2 get i * get global Graph_Resolution
    y2 select list item list get global Graph_Y_Val_List
    index get i + 1
  set CurPos Val lbl to get global Graph_Current X_Pos
```



```

when START_BTN.Click
do
  if compare texts START_BTN.Text = PAUSE
  then
    set Clock1.TimerEnabled to false
    set START_BTN.Text to START
    set STEP_BTN.Enabled to true
  else
    set Clock1.TimerEnabled to true
    set START_BTN.Text to PAUSE
    set STEP_BTN.Enabled to false
  end if
end do

```

```

when RESET_BTN.Click
do
  call reset
end do

```

```

when Screen1.ScreenOrientationChanged
do
  set global comment to join
  " A timer is used because screen width does not up..."
  " fast enough when the orientation is changed."
  " This means without a timer maxPos is always one ..."
  " behind the actual orientation"
  set Clock2.TimerEnabled to true
end do

```

```

when Clock2.Timer
do
  set Clock2.TimerEnabled to false
  call GraphChangeOrientation
end do

```

```

when Clock1.Timer
do
  set Clock1.TimerEnabled to false
  if 0 and BluetoothClient1.isConnected
  then
    call BluetoothClient1.BytesAvailableToReceive > 0
  then
    set global value to call BluetoothClient1.ReceiveText
    numberOfBytes - 1
  end if
  call updateGraph
  set Clock1.TimerEnabled to true
end do

```

```

when STEP_BTN.Click
do
  set global value to random integer from get global minVal to get global maxVal
  call updateGraph
end do

```

```

to mapValue val inSTART inEND outSTART outEND
result
do
  initialize local inRange to get inEND - get inSTART
  initialize local outRange to get outEND - get outSTART
  initialize local temp1 to 0
  initialize local temp2 to 0
  in
  set val to get val
  set temp1 to get outRange / get inRange
  set temp2 to get temp1 * get outSTART
  set val to round get val * get temp2
end in
result get val
end to

```

```

to GraphChangeOrientation
do
  set global timerStatus to Clock1.TimerEnabled
  set Clock1.TimerEnabled to false
  call Init_Graph
  if get global Graph_Current_X_Pos > get global Graph_maxPos
  then
    initialize local diff2 to get global Graph_Current_X_Pos - get global Graph_maxPos
    initialize local number to 1
    in while test get number <= get global Graph_maxPos
    do
      replace list item list get global Graph_Y_Val_List
      index get number
      replacement select list item list get global Graph_Y_Val_List
      index 0 get number + get diff2
      set number to get number + 1
    end in
    set global Graph_Current_X_Pos to get global Graph_maxPos
  end if
  if not get global timerStatus
  then
    call updateGraph
  end if
  set Clock1.TimerEnabled to get global timerStatus
end do

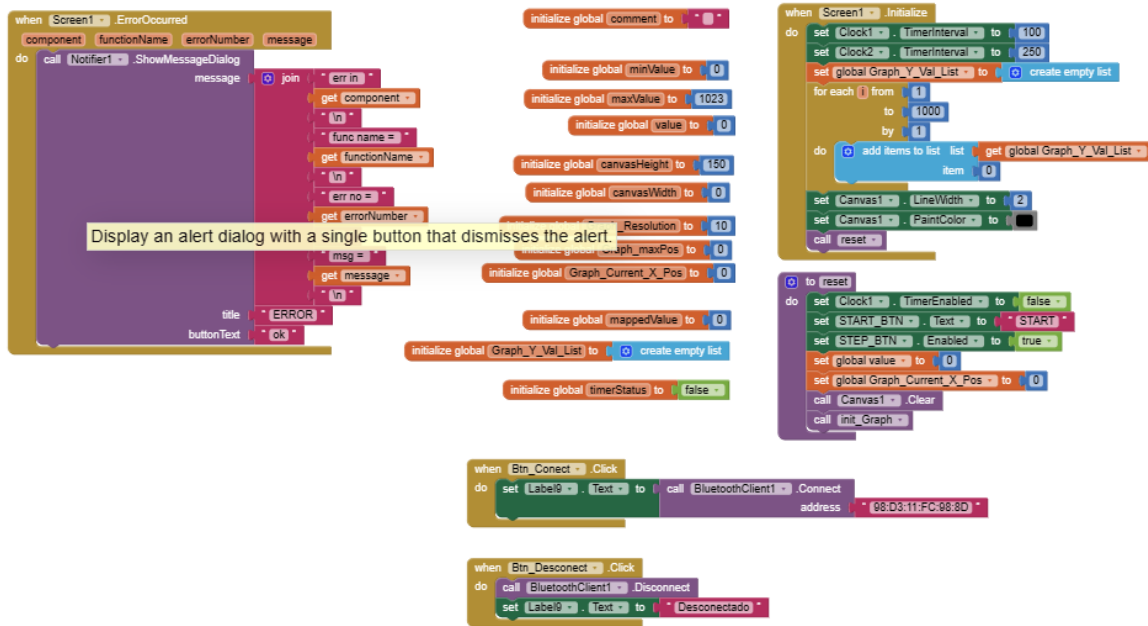
```

```

to Init_Graph
do
  if Screen1.Height > Screen1.Width
  then
    set bl_SPACER_01.Height to 15
    set bl_SPACER_02.Height to 10
    set global canvasWidth to Canvas1.Width
    set Canvas1.Height to get global canvasHeight
    set Canvas1.BackgroundImage to Grid_350x120_10_OL.png
  else
    set bl_SPACER_01.Height to 6
    set bl_SPACER_02.Height to 6
    set global canvasWidth to Canvas1.Width
    set Canvas1.Height to get global canvasHeight
    set Canvas1.BackgroundImage to Grid_550x120_10_OL.png
  end if
  set global Graph_maxPos to round get global canvasWidth / get global Graph_Resolution
  set global comment to " If the last value goes past the end of the graph..."
  set global Graph_maxPos to get global Graph_maxPos - 1
  set canvasSize_Val lbl to join W= Canvas1.Width
  H= Canvas1.Height
  set maxPos_Val lbl to get global Graph_maxPos
  set xScale_Val lbl to get global Graph_Resolution
  set curPos_Val lbl to get global Graph_Current_X_Pos
end do

```





The Blocks Editor in MIT App Inventor is where users add functionality to their app by dragging and dropping code blocks. The Blocks Editor provides a visual representation of the app's logic, allowing users to create complex functionality without having to write any code. Users can drag blocks that represent programming concepts such as loops, variables, and functions and snap them together to create the app's functionality. The Blocks Editor also includes pre-built blocks for accessing device features such as the camera or GPS location, making it easy to add these features to an app.

In MIT App Inventor, the Blocks Editor is an essential tool for creating the behavior of your user interface components. Blocks are code snippets that define how your components respond to user input, such as button clicks, text input, and other events. You can use blocks to create complex functionality, such as data storage, network communication, and media playback. The Blocks Editor is a drag-and-drop interface that allows you to create and connect blocks easily.

To access the Blocks Editor in MIT App Inventor, select the "Blocks" tab in the top left corner of the screen. From there, you can choose from a variety of categories, including "Control," "Logic," "Math," and "Text," among others. Each category contains blocks that you can use to create your desired functionality.

For instance, you can use the "Control" category to create blocks that define the flow of your program. You can use the "Logic" category to create blocks that define conditions and operations, such as "If/else," "And/Or," and "Not." You can use the "Math" category to create blocks that perform mathematical operations, such as addition, subtraction, multiplication, and division. You can use the "Text" category to create blocks that manipulate text, such as joining, splitting, and formatting.

Once you have selected the desired block, you can drag it onto the workspace and connect it to other blocks to create your desired functionality. For instance, you can create a block that defines the action to be taken when a button is clicked. You can then connect that block to other blocks that define the behavior of other components in your user interface.

Overall, the Blocks Editor in MIT App Inventor is a powerful tool that allows you to create complex functionality with ease. With a little creativity and some knowledge of programming concepts, you can create a user interface that meets your needs and engages your users. If you encounter any issues while using the Blocks Editor, you can refer to the MIT App Inventor documentation or seek help from the community forums. There are also many online tutorials and resources available to help you get started and improve your skills. Good luck with your project!

In addition to the categories mentioned earlier, there are other categories available in the Blocks Editor that you can use to create more complex functionality in your app. These categories include "Lists," "Variables," "Procedures," and "Components."

The "Lists" category allows you to create and manipulate lists of data. For example, you can use this category to create a list of items that are displayed in a list view component. You can also use this category to sort and filter the data in your lists.

The "Variables" category allows you to create and manipulate variables, which are used to store data in your app. You can use variables to store user input, calculated values, and other types of data. You can also use variables to pass data between different components in your app.

The "Procedures" category allows you to create custom blocks that can be reused throughout your app. For example, you can create a custom block that performs a specific action, such as sending a text message or updating a database. You can then use this block in different parts of your app, making your code more modular and easier to maintain.

The "Components" category allows you to access the properties and methods of the components in your app. By using this category, you can customize the behavior of your components and add new features to your app. For example, you can use this category to add animation effects to your user interface components.

Using these categories, you can create more complex and powerful apps in MIT App Inventor. If you're new to programming or app development, it may take some time to get used to the Blocks Editor and its various categories. However, with practice and patience, you can become proficient in using this tool to create amazing apps.

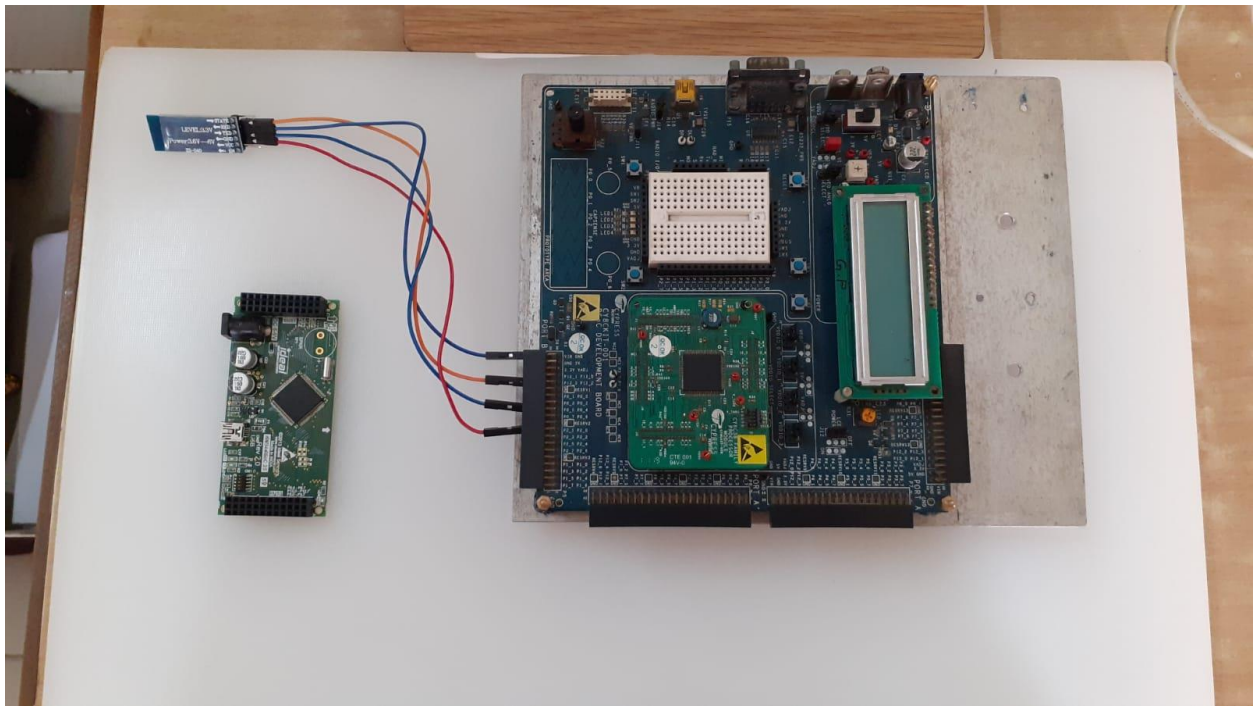
CHAPTER 5

RESULT

Result from Bluetooth module

A Bluetooth module is inserted to the PSoC and results are obtained in MIT App through which the graphs are observed and sample values are received . From the sampling values sampling will be calculated. The results can be observed in smart devices.

Figure 17: Insertion of Bluetooth module



Primarily the module is connected to a development board and later on it is to be connected to a wearable system glove.

Result from power backup

After the Bluetooth insertion the power backup tool is mounted to the wearable glove and hence it can also make it low power consumption kit and from the battery attached to it can be rechargeable whenever needed.

Figure 18: PSoC board & Power backup tool

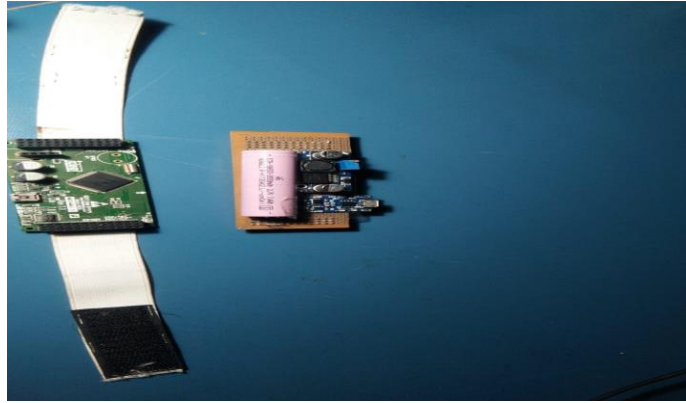
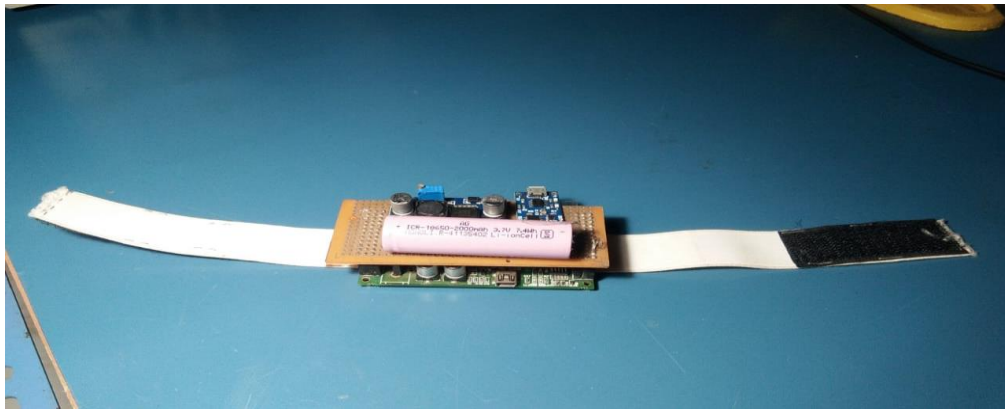


Figure 19: Power module with sensor and bluetooth module mounted on PSoC module

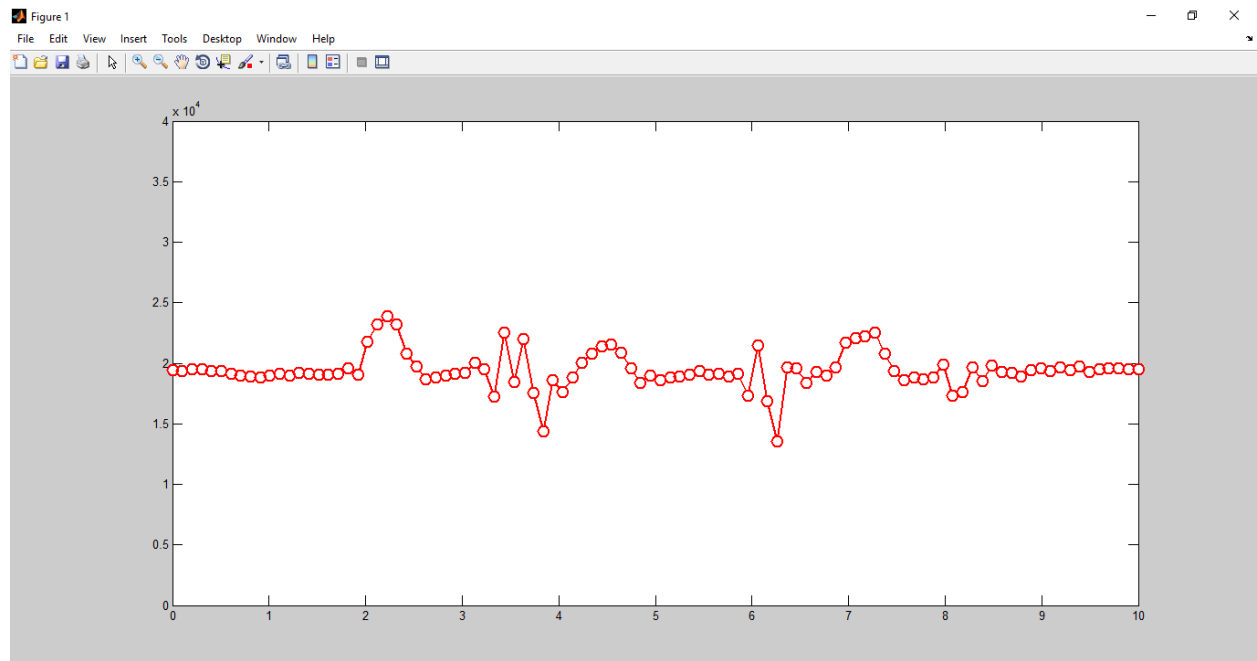


Results for sampling rate

From the matlab lab usage a acceleration graph is obtained and sampling rate is calculated from the number of sampling values per second.

With an accelerometer sensor connected the PSoC board and through an usb cable or by wireless communication(Bluetooth module) the Matlab graph is gained and this graph can also be observed in MIT app but using other evaluation methods.

Figure 20: Matlab output



CONCLUSION

Finally a wearable device is designed with the help of an ADXL-335 sensor, HC-05 module and a power handler. From this making a portable device, enabling the power management capability, receiving the sampling rate from the smart device is obtained successfully. From this few of the limitations have been cleared and made the device human friendly.

However more future improvements are to be done to this device to make it self monitoring and also enhancing the samling rate need to be done.

APPENDIX

1 PSOC program and description

PSOC KEIL-C PROGRAM

```
#include<project.h>
#include "stdio.h"

#ifdef (__GNUC__)
/*to use the floating point conversion specifiers*/
/*Add an explicit reference to the floating point printf library. */
/*This is not by default linked in with the newlib-nano library.*/
asm (".global _printf_float");
#endif

#define USBFS_DEVICE (0u)

/* The buffer size is equal to the maximum packet size of the IN and OUT bulk
 * Endpoints.
 */
#define USBUART_BUFFER_SIZE (64u)
#define LINE_STR_LENGTH (20u)

char8* parity[] = {"None", "Odd", "Even", "Mark", "Space"};
char8* stop[] = {"1", "1.5", "2"};
char * Accelval[6];
uint16sensorout;

/*****
**
* Function Name: main
*****
*
* The main function carries out the following tasks:
* 1. It waits for VBUS to become valid before launching the USBFS component, which is
listed as a virtual Com port.
* 2. Waits for the device to be enumerated by the host.
* 3. Waits for data flowing from the hyper terminal and sends it back
* 4. PsoC3/PsoC5LP: the LCD displays the line settings.
*****
*/
intmain()
{
uint16 count;
uint8buffer[USBUART_BUFFER_SIZE];
```



```

//uint8 flag=0;

#if (CY_PSOC3 || CY_PSOC5LP)
uint8 state;
char8lineStr[LINE_STR_LENGTH];

//LCD_Start();
  ADC_DelSig_1_Start();
#endif/* (CY_PSOC3 || CY_PSOC5LP) */

CyGlobalIntEnable;

/* Start USBFS operation with 5-V operation. */
USBUART_Start(USBFS_DEVICE, USBUART_5V_OPERATION);

for(;;)
{

/* Host can send double SET_INTERFACE request. */
if (0u != USBUART_IsConfigurationChanged())
{
/* Initialize IN endpoints when device is configured. */
if (0u != USBUART_GetConfiguration())
{
/* Enumeration is done, enable OUT endpoint to receive data
   * from host. */
USBUART_CDC_Init();
}
}

/* Service USB CDC when device is configured. */
if (0u != USBUART_GetConfiguration())
{
/* Check for input data from host. */
if (0u != USBUART_DataIsReady())
{
/* Read received data and re-enable OUT endpoint. */
count = USBUART_GetAll(buffer);
ADC_DelSig_1_StartConvert();
if(ADC_DelSig_1_IsEndConversion(ADC_DelSig_1_WAIT_FOR_RESULT))
{
sensorout=ADC_DelSig_1_GetResult32();//read the numerical analog value
48print((char *)Accelval,"%u",sensorout);//prepare an ascii string to send over usb
strcat(Accelval,"\n");
}
}
}
}

```

```

    }

if (0u != count)
    {
/* Wait until component is ready to send data to host. */
while (0u == USBUART_CDCIsReady())
    {
    }
USBUART_PutString(Accelval);

/* If the last sent packet is exactly the maximum packet
   * size, it is followed by a zero-length packet to assure
   * that the end of the segment is properly identified by
   * the terminal.
   */
if (USBUART_BUFFER_SIZE == count)
    {
/* Wait until component is ready to send data to PC. */
while (0u == USBUART_CDCIsReady())
    {
    }

/* Send zero-length packet to PC. */
USBUART_PutData(NULL, 0u);
    }
    }

/* [] END OF FILE */

```

4.1.3 PSOC PROGRAM COMMANDS DESCRIPTION

i. USBUART_PutString

Sends a RAM string with a null termination to the computer. When sending huge packets, go to Sending Packets of Length 64 Bytes and More.

C Prototype:

```
void USBUART_PutString(BYTE * pStr)
```

Assembler:

move A,>pStr; load the MSB portion of the pointer to the RAM-based null; terminate the string loading the LSB portion of a reference to a RAM-based null; terminating the string

Call the function to send the string using `lcallUSBUART_PutString`.

Parameters:

`pStr`: Pointer to the PC-to-be-sent string. The Accumulator receives the MSB, while the X register receives the LSB. 64 bytes is the maximum string length, including the final null character.

Return Value:

None

Side Effects:

This version of the function, as well as any future ones, may change the A and X registers. In the huge memory paradigm, this is true for every RAM page pointer register. The calling function must, where necessary, maintain the values across calls to `fastcall16` functions. Only the `IDX_PP` and `CUR_PP` page pointer registers are changed right now.

ii. Unit 8 USBUART_GetConfiguration(void)

Description:

This function retrieves the USB device's

Return Value:

`uint8` current configuration: returns the configuration that is presently allocated. If the device is not configured, it returns 0.

iii. uint8 USBUART_Is Configuration Changed(void)

Description:

The clear-on-read configuration state is what this function returns. When the host sends two `SET_CONFIGURATION` requests with the same configuration number or modifies the interface's alternate settings, it can be helpful.

To begin communication with the host after configuration changes have been made, the OUT endpoints must be enabled and the IN endpoints must be loaded with data.

Return Value:

`uint8`: If a new configuration has been modified, it returns a nonzero value otherwise it returns zero.

iv. void USBUART_CDC_Init(void)

Description:

The CDC interface is initialised by this function so that it is prepared to accept data from the PC. In the case of multiple communication port support, the API set the current communication port to 0. To initialise and launch the USBFS Component operation, this API should be used after the

device has been launched and configured using the USBUART_Start() API. Once the host has configured and enumerated the device, call the USBUART_GetConfiguration() API to wait.

For example:

```
USBUART_Start(...);
while(0 == USBUART_GetConfiguration())
{
}
USBUART_CDC_Init();
```

Parameters: None

Return Value:None

Side Effects: None

v. uint8 USBUART_DataIsReady(void)

Description:

If the Component receives data or a packet with zero length, this function returns a nonzero result. Even when a zero-length packet is received, the USBUART_GetAll() or USBUART_GetData() API should be used to extract data from the buffer and reinitialize the OUT endpoint. When a packet with zero length is received, these APIs will return zero.

Parameters: None

Return Value:

uint8: This method returns a nonzero value if the OUT packet is successfully received. It yields 0 in all other cases.

Side Effects:None

vi. uint8 USBUART_IsLineChanged(void)

Description:

The clear-on-read status of the line is returned by this function. When the host transmits the device updated coding or control information, it does not return a value of zero. To read data coding information, the USBUART_GetDTERate(), USBUART_GetCharFormat(), USBUART_GetParityType(), or USBUART_GetDataBits() API should be used. To read line control information, use the USBUART_GetLineControl() API.

Parameters: None

Return Value:

uint8: It returns a nonzero value if SET_LINE_CODING or CDC_SET_CONTROL_LINE_STATE requests are received. It yields 0 in all other cases

Return Value**Description**

USBUART_LINE_CODING_CHANGE Line coding changed
D

USBUART_LINE_CONTROL_CHAN Line control changed
GED

Side Effects: None

vii. uint8 USBUART_GetParityType(void)

Description: This function returns the parity type for the CDC port.

Parameters: None

Return Value: unit 8

Return Value**Description**

USBUART_PARITY_NONE None

USBUART_PARITY_ODD Odd

USBUART_PARITY_EVEN Even

USBUART_PARITY_MARK Mark

USBUART_PARITY_SPACE Space

viii. uint8 USBUART_GetDataBits(void)

Description: This function returns the number of data bits for the CDC port

Parameters: None

Return Value:

Returns the number of data bits in uint8 format. Data bits might range from 5, 6, 7, 8, or 16 in number.

Side Effects: None

ix. uint16 USBUART_GetLineControl(void)

Description: This function returns the line control bitmap that the host sends to the device.

Parameters: None.

Return Value: uint8:

Return Value	Notes
USBUART_LINE_CONTROL_DTR	shows the presence of a DTR signal. This signal is the same as RS232 signal DTR and V.24 signal 108/2.
USBUART_LINE_CONTROL_RTS	Half-duplex modem carrier control. This signal is the same as RS232 signal RTS and V.24 signal 105.
RESERVED	The remaining components are set aside.

Side Effects : None

x. uint16 USBUART_GetAll(uint8* pData)

Description:

This function pulls every byte of data that has been received from the input buffer and puts them into a designated data array. To confirm that data has been received from the host, the USBUART_DataIsReady() API should be called first.

Parameters: uint8* pData: Pointer to the data array in which the data will be stored.

Return Value:

uint16: The quantity of data received. The most data that can be received at once is 64 bytes.

Side Effects: None

xi. uint32 USBUART_GetDTERate(void)

Description: The data terminal rate configured for this port by this function is returned in bits per second.

Parameters: None

Return Value: uint32: Provides the data rate in bits per second as a value.

Side Effects: None

xii. uint8 USBUART_GetCharFormat(void)

Description: This function returns the number of stop bits.

Parameters: None

Return Value: uint8: Returns the number of stop bits.

Description

USBUART_1_STOPBIT 1 stop bit
 USBUART_1_5_STOPBIT 1,5 stop bits
 S
 USBUART_2_STOPBITS 2 stop bits

Side Effects: None

xiii. USBUART_Start

Description:

carries out every action necessary to launch the USBUART Device User Module. The chip's operational voltage, transferred through the accumulator, is indicated by the parameter b Voltage. This decides whether pass through mode is employed for 3.3 V operation or the voltage regulator is engaged for 5 V operation. The following table lists the symbolic names that are provided in C and assembly, along with the values that go along with them.

Mask	Value	Description
USBUART_3V_OPERATION	0x02	Pass through Vcc for pull up while disabling the voltage regulator.
USBUART_5V_OPERATION	0x03	Turn on the voltage regulator and use the pull-up function of the regulator.

Table 2 : parameters of USBUART_Start command

Return Value: None

Side Effects:

The A and X registers are capable of being changed by this function or subsequent iterations. In the huge memory paradigm, this is true for every RAM page pointer register. The calling function must, where necessary, maintain the values across calls to fastcall16 functions. Only the IDX_PP and CUR_PP page pointer registers are changed right now.

xiv. void LCD_Char_Position(uint8 row, uint8 column)

Description: Moves the cursor to the location specified by arguments **row** and **column**.

Parameters: uint8 row: The cursor's current location in the row. The smallest value is zero.
 Uint8 column: The cursor's current location in the column. 0 is the minimum value.

Return Value: None

Side Effects: None

xv. void LCD_Char_PrintString(char8 const string[])

Description: begins at the current cursor position and writes a string of characters to the screen that is null-terminated.

Parameters: For the LCD module's display: char8 conststring[]: a null-terminated array of ASCII characters.

Return Value: None

Side Effects: None

xvi. void LCD_Char_Start(void)

Description:

The LCD hardware module is initialised by this function as follows:

- The cursor is reset to its initial location
- the 4-bit interface is enabled
- the display is cleared, and auto cursor increment is enabled.

Additionally, if a custom character set was established in the customizer's GUI, it loads it onto the LCD.

Parameters: None

Return Value: None

Side Effects: None

xvii. void ADC_Start(void)

Description:

invokes the ADC_Init() method, sets the initVar variable, and then the ADC_Enable() function.

The ADC is configured and turned on by this function, but conversions are not started. The ADC is set up by default for Config1. After that, choose a different configuration using the ADC_SelectConfiguration() method.

Side Effects:

allows for internal interruptions. The interruption is ended by reading the outcome. For instructions on how to remove an internal interrupt, see the section on Interrupt Service Routines.

xviii. void ADC_StartConvert(void)

Description:

Makes the ADC start a conversion by force. To begin a single conversion while in Single Sample mode, contact this API. Use the ADC_IsEndConversion() API to verify or wait for this event to occur, at which point the ADC will halt. A new conversion will begin once the current conversion is completed if the ADC_StartConvert() method is called while the conversion is still

in progress. Stop the current conversion by executing ADC_StopConvert if you wish to begin a new conversion without waiting for the previous one to complete.(). Call ADC_StartConvert to resume the conversion once it has been stopped.().Calling this API will initiate continuous ADC conversions in Multi Sample, Continuous, or Multi Sample (Turbo) modes until the ADC_StopConvert() or ADC_Stop() functions are called.

xix. uint8 ADC_IsEndConversion(uint8 retMode)

Description:

Examines the ADC end of the conversion. The programmer has two alternatives when using this function. This function returns the conversion status quickly in one mode. In the alternative option, the function blocks and does not return until the conversion is finished.

Parameters:

uint8 ret Mode: Check conversion return mode. See the following table for options.

Options	Description
ADC_RETURN_STATUS	Immediately returns conversion result status.
ADC_WAIT_FOR_RESULT	Does not return until ADC conversion is complete.

Return Value:

uint8: The final conversion has finished if a nonzero value is returned. If the value that was returned was zero, the ADC was still working on the previous result.

Side Effects:

The ADC_IsEndConversion() method should not be used to signal the DMA to read the ADC result when the EOC output is being used. This is so that this API may not return when in the ADC_WAIT_FOR_RESULT mode since reading the output register with DMA clears the ADC's conversion complete flag.

xx. int32 ADC_GetResult32(void)

Description:

returns a result with a 32-bit size for a conversion with an 8–20 bit resolution.

Return Value:

int32: Result of the last ADC conversion.

2 Matlab program and design

MATLAB PROGRAM

```
clc
clear all
% s = serial('COM8','baudrate',9600);
% s = serial('COM5','baudrate',57600);
s = serial('COM8','baudrate',9600);
set(s, 'InputBufferSize', 30);
set(s,'Terminator', 'LF');

s.BytesAvailableFcnMode='terminator';
%s.BytesAvailableFcn=@VentRead;
fopen(s);
pause on;

x = linspace(0,10,100);
k=length(x);
waveform=zeros(1,10000);
% y = sin(x);
y=zeros(1,k);
figure(1);
% h = plot(x,y);
h= plot(x,y,'-ro','LineWidth',2,...
'MarkerEdgeColor','r',...
'MarkerFaceColor','w',...
'MarkerSize',10);
set(h,'XdataSource','x');
set(h,'YdataSource','y');
ylim([0, 40000]);

spectra=fft(y);

figure(2);
j= plot(x,spectra,'-ko','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerFaceColor','w',...
'MarkerSize',5);
set(j,'XdataSource','x');
set(j,'YdataSource','spectra');
ylim([-1000, 100000]);
xlim([0.25,5]);
```

```

% y = sin(x.^3);
for i=1:100
    fprintf(s,'s');
    % pause(0.1);
    % b='4193';
    b=fgets(s);

    y(k)=str2double(b);

    waveform(i)=y(k);

refreshdata(1)
% pause(0.1);
y = circshift(y,[0,-1]);
spectra=fft(y);
refreshdata(2)

flushinput(s);
pause(0.01);

end
fclose(s);

```

4.2.4 MATLAB PROGRAM COMMANDS DESCRIPTION

SERIAL OBJECT

The SERIAL - OBJECT idea serves as the programme's foundation.

Create a serial port object with the serial command.

The instrcb wrapper is used for serial object callbacks.

Find communication interface objects with the supplied property values using the instrfind command.

Find all communication interface objects with the provided criteria using instrfindall

Both a directory and a function, serial.

construct a serial port object in serial.

A serial port object $S = \text{serial}(\text{'PORT'})$ is created and linked to the port, PORT. You won't be able to attach the serial port object to the device if PORT doesn't exist or is already in use.

The object must be attached to the serial port using the FOPEN function in order to communicate with the device.

The Status property of the serial port object is closed during construction. The Status property is set to open after the object is attached to the serial port using the FOPEN function. A serial port can only have one device attached to it at once.

S = serial('PORT','P1','V1','P2','V2,...) creates a serial port object with the supplied property values that is associated with the port, PORT. The object won't be generated if an invalid property name or property value is provided.

It should be noted that the property value pairs may be in any of the SET function's supported formats, including param-value string pairs, structures, and param-value cell array pairs.

Example:

```
% To construct a serial port object:
s1 = serial('COM1');
s2 = serial('COM2', 'BaudRate', 1200);

% To connect the serial port object to the serial port:
fopen(s1)
fopen(s2)

% To query the device.
fprintf(s1, '*IDN?');
idn = fscanf(s1);

% To disconnect the serial port object from the serial port.
fclose(s1);
fclose(s2);
```

FOPEN

fopen Connect the device to the serial port object.

The serial port object, OBJ, is connected to the gadget via the function fopen(OBJ). An array of serial port objects could make up OBJ.

A single object with the same configuration of a serial port can only be attached to an instrument at once. One serial port object, for instance, can only be linked to the COM2 port at once. If OBJ linked to the device successfully, the Status property of OBJ is set to open; otherwise, it is left set to closed.

Any data left in the input and output buffers is flushed when the OBJ is opened, and the BytesAvailable, BytesToOutput, Values Received, and Values Sent properties are all reset to 0.

Only once the device has been connected may some property values be confirmed. BaudRate, Flow Control, and Parity are a few examples. An error will be given and the object won't be linked to the device if any of these properties are set to a value that the device does not support.

Before using fopen, some properties must be specified because they are read-only while the serial port object is open (attached). InputBufferSize and OutputBufferSize are two examples.

If fopen is used on a serial port object with the Status property set to Open, an error will be returned.

The OBJ's ByteOrder field allows users to specify the device's byte order.

The remaining objects in the array will connect to the device and a warning will be displayed if OBJ is an array of serial port objects and one of the objects cannot be connected to the device.

Example:

```
s = serial('COM1');
fopen(s);
fprintf(s, '*IDN?');
idn = fscanf(s);
fclose(s);
```

FCLOSE

fclose Shut the file.

The command `ST = fclose(FID)` shuts the associated file. FID is an integer number received from a previous call to FOPEN. If successful, Fclose returns 0; otherwise, it returns -1. Fclose throws an error if FID does not indicate an open file or if it equals 0 (standard input), 1 (standard output), or 2 (standard error).

Except for 0 and 1, all open files are closed by `ST = fclose('all')`.

clc-Clear command window.

An N-by-N zeros matrix is known as Zeros(N).

Clc homes the cursor and clears the command window.

Get functions and variables out of your memory.

The workspace is cleared of all variables.

This is what Clear VARIABLES accomplishes.

All global variables are eliminated using clear GLOBAL.

All compiled MATLAB and MEX-functions are removed with Clear FUNCTIONS.

Calling `clean` FUNCTIONS is typically redundant and reduces code performance.

See the page marked "Clear Reference" for more details.

All variables, globals, functions, and MEX connections are deleted by clearing `ALL`.

At the command prompt, typing `clean ALL` also clears the base import list.

`Clear ALL` is typically unnecessary and reduces code performance.

See the page marked "Clear Reference" for more details.

linspace Linearly spaced vector.

A row vector of 100 linearly equal-spaced points between `X1` and `X2` is produced by the function `Linspace(X1, X2)`.

`N` points between `X1` and `X2` are produced by `Linspace(X1, X2, N)`.

`Linspace` returns `X2` when `N = 1`.

zeros Zeros array.

An `N`-by-`N` zeros matrix is known as `Zeros(N)`.

An `M` by `N` matrix of zeros is known as `zeros (M, N)` or `zeros ([M, N])`.

`Zeros (M, N, P,...)` or `zeros([M N P...])` is an array of zeros with the dimensions `M` by `N` by `P` by...

All zeros (`SIZE(A)`) have the same size as `A`.

The scalar is zeros with no parameters. `0`

`Zeros (...)` is an array of zeros belonging to the class denoted by the string `CLASSNAME`.

An array of zeros with the same data type, sparsity, and complexity (real or complex) as the numeric value `Y` is called `Zeros (..., "like," Y)`.

For instance, `x = zeros(2,3,'int8');`

figure Create figure window.

When used alone, `Figure` generates a fresh figure window and returns its handle.

`H` is elevated above all other figures on the screen by `Figure(H)`, which also forces `H` to become visible. If `H` is an integer and `Figure H` does not already exist, a new figure with handle `H` is produced.

The handle to the current figure is returned by GCF.

To view a list of figure properties and their current values, run GET(H). To examine a list of a figure's properties and potential values, run SET(H).

PAUSE WAIT FOR USER RESPONSE.

Pause(n), where n may also be a fraction, stops for n seconds before resuming. Platform particularity governs the clock's resolution. On most platforms, 0.01 second fractional pauses should be supported.

A procedure will pause and wait till the user presses any key before proceeding.

When you specify Pause OFF, any pause or pause(n) commands after that shouldn't actually pause. Normally interactive scripts can now be run in an unattended manner.

Following pause instructions should pause if Pause ON is present.

Pause QUERY returns the status of the pause at the moment, either "on" or "off."

Prior to processing the input arguments, STATE = pause(...) returns the state of the pause.

The operating system you are using's scheduling resolution as well as other system activity will affect how accurate pause is. It can only be guaranteed with some predicted mistake; it cannot be guaranteed with 100% confidence. Choosing N with a resolution of.1 (such as N = 1.7), for instance, produces actual pause values that are accurate to about 10% in the relative inaccuracy of the fractional part, according to studies. Higher relative inaccuracy is revealed when asking for finer resolutions (like.01).

REFERENCES

- [1] Rovini, Erika, Carlo Maremmani, and Filippo Cavallo. "How wearable sensors can support Parkinson's disease diagnosis and treatment: a systematic review." *Frontiers in neuroscience* 11 (2017): 555.
- [2] Alhamid, M. F., Alamri, A., and El Saddik, A. (2010). "Measuring hand-arm steadiness for post-stroke and Parkinson's disease patients using SIERRA framework," in *International Workshop on Medical Measurements and Applications (MeMeA)* (Ottawa, ON: IEEE), 6–9.
- [3] Bächlin, Marc, et al. "Parkinson's disease patients perspective on context aware wearable technology for auditive assistance." *2009 3rd International Conference on Pervasive Computing Technologies for Healthcare*. IEEE, 2009.
- [4] Cavallo, F., Esposito, D., Rovini, E., Aquilano, M., Carrozza, M. C., Dario, P., ... & Bongioanni, P. (2013, June). "Preliminary evaluation of SensHand V1 in assessing motor skills performance in Parkinson disease." In *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)* (pp. 1-6). IEEE.
- [5] Vinjamuri, Ramana, et al. "Extraction of sources of tremor in hand movements of patients with movement disorders." *IEEE Transactions on Information Technology in Biomedicine* 13.1 (2008): 49-56.
- [6] Niazmand, K., Tonn, K., Kalaras, A., Fietzek, U. M., Mehrkens, J. H., & Lueth, T. C. (2011, June). Quantitative evaluation of Parkinson's disease using sensor based smart glove. In *2011 24th International Symposium on Computer-Based Medical Systems (CBMS)* (pp. 1-8). IEEE.
- [7] Kostikis, N., Hristu-Varsakelis, D., Arnaoutoglou, M., & Kotsavasiloglou, C. (2015). A smartphone-based tool for assessing parkinsonian hand tremor. *IEEE journal of biomedical and health informatics*, 19(6), 1835-1842.
- [8] Patel, S., Lorincz, K., Hughes, R., Huggins, N., Growdon, J., Standaert, D., ... & Bonato, P. (2009). Monitoring motor fluctuations in patients with Parkinson's disease using wearable sensors. *IEEE transactions on information technology in biomedicine*, 13(6), 864-873.
- [9] Weiss, A., Sharifi, S., Plotnik, M., van Vugt, J. P., Giladi, N., & Hausdorff, J. M. (2011). Toward automated, at-home assessment of mobility among patients with Parkinson disease, using a body-worn accelerometer. *Neurorehabilitation and neural repair*, 25(9), 810-818.
- [10] Rigas, G., Tzallas, A. T., Tsipouras, M. G., Bougia, P., Tripoliti, E. E., Baga, D., ... & Konitsiotis, S. (2012). Assessment of tremor activity in the Parkinson's disease using a set of wearable sensors. *IEEE Transactions on Information Technology in Biomedicine*, 16(3), 478-487.

- [11] Wenzelburger, R., Raethjen, J., Löffler, K., Stolze, H., Illert, M., & Deuschl, G. (2000). Kinetic tremor in a reach-to-grasp movement in Parkinson's disease. *Movement disorders: official journal of the Movement Disorder Society*, 15(6), 1084-1094.
- [12] Ward, J. A., Lukowicz, P., Troster, G., & Starner, T. E. (2006). Activity recognition of assembly tasks using body-worn microphones and accelerometers. *IEEE transactions on pattern analysis and machine intelligence*, 28(10), 1553-1567.
- [13] Mariani, B., Jiménez, M. C., Vingerhoets, F. J., & Aminian, K. (2012). On-shoe wearable sensors for gait and turning assessment of patients with Parkinson's disease. *IEEE transactions on biomedical engineering*, 60(1), 155-158.
- [14] Rocchi, Laura, L. Chiari, and FB1738049 Horak. "Effects of deep brain stimulation and levodopa on postural sway in Parkinson's disease." *Journal of Neurology, Neurosurgery & Psychiatry* 73.3 (2002): 267-274.
- [15] Macleod, Angus D., and Carl E. Counsell. "Timed tests of motor function in Parkinson's disease." *Parkinsonism & related disorders* 16, no. 7 (2010): 442-446.
- [16] Caldara, M., Comotti, D., Galizzi, M., Locatelli, P., Re, V., Alimonti, D., ... & Rizzetti, M. C. (2014, June). A novel body sensor network for Parkinson's disease patients rehabilitation assessment. In *2014 11th International Conference on Wearable and Implantable Body Sensor Networks* (pp. 81-86). IEEE.
- [17] Frazzitta, G., Balbi, P., Maestri, R., Bertotti, G., Boveri, N., & Pezzoli, G. (2013). The beneficial role of intensive exercise on Parkinson disease progression. *American journal of physical medicine & rehabilitation*, 92(6), 523-532.
- [18] Comotti, D., Ermidoro, M., Galizzi, M., & Vitali, A. (2013, May). Development of a wireless low-power multi-sensor network for motion tracking applications. In *2013 IEEE International Conference on Body Sensor Networks* (pp. 1-6). IEEE.
- [19] Morris, T. R., Cho, C., Dilda, V., Shine, J. M., Naismith, S. L., Lewis, S. J., & Moore, S. T. (2013). Clinical assessment of freezing of gait in Parkinson's disease from computer-generated animation. *Gait & posture*, 38(2), 326-329.
- [20] Moore, S. T., MacDougall, H. G., Gracies, J. M., Cohen, H. S., & Ondo, W. G. (2007). Long-term monitoring of gait in Parkinson's disease. *Gait & posture*, 26(2), 200-207.
- [21] Morris, T. R., Cho, C., Dilda, V., Shine, J. M., Naismith, S. L., Lewis, S. J., & Moore, S. T. (2012). A comparison of clinical and objective measures of freezing of gait in Parkinson's disease. *Parkinsonism & related disorders*, 18(5), 572-577.

- [22] Mellone, S., Palmerini, L., Cappello, A., & Chiari, L. (2011). Hilbert–Huang-based tremor removal to assess postural properties from accelerometers. *IEEE transactions on biomedical engineering*, 58(6), 1752-1761.
- [23] Hellwig, B., Mund, P., Schelter, B., Guschlbauer, B., Timmer, J., & Lücking, C. H. (2009). A longitudinal study of tremor frequencies in Parkinson’s disease and essential tremor. *Clinical Neurophysiology*, 120(2), 431-435.
- [24] Streepey, J. W., Kenyon, R. V., & Keshner, E. A. (2007). Field of view and base of support width influence postural responses to visual stimuli during quiet stance. *Gait & posture*, 25(1), 49-55.
- [25] Connolly, E. S(2014) , Appelboom, G., Camacho, E., Abraham, M. E., Bruce, S. S., Dumont, E. L., Zacharia, B. E.,. “Smart wearable body sensors for patient self-assessment and monitoring”. *Archives of public health*, 72(1), 1-9.
- [26] Wu, Y. C., Chen, P. F., Hu, Z. H., Chang, C. H., Lee, G. C., & Yu, W. C. (2009, December). A mobile health monitoring system using RFID ring-type pulse sensor. In *2009 Eighth IEEE International conference on dependable, autonomic and secure computing* (pp. 317-322). IEEE.
- [27] Hayakawa, M., Uchimura, Y., Omae, K., Waki, K., Fujita, H., & Ohe, K. (2013). A smartphone-based medication self-management system with real-time medication monitoring. *Applied clinical informatics*, 4(01), 37-52.
- [28] Morgan, Catherine, Michal Rolinski, Roisin McNaney, Bennet Jones, Lynn Rochester, Walter Maetzler, Ian Craddock, and Alan L. Whone. "Systematic review looking at the use of technology to measure free-living symptom and activity outcomes in Parkinson’s disease in the home or a home-like environment." *Journal of Parkinson's disease* 10, no. 2 (2020): 429-454.
- [29] Post B , Merkus MP , de Bie RMA , de Haan RJ , Speelman JD ((2005)) Unified Parkinson’s disease rating scale motor examination: Are ratings of nurses, residents in neurology, and movement disorders specialists interchangeable? *Mov Disord* 20: , 1577–84.
- [30] Gao J , Tian F , Fan J , Wang D , Fan X , Zhu Y , Ma S , Huang J , Wang H ((2018)) Implicit detection of motor impairment in Parkinson’s disease from everyday smartphone interactions. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems - CHI ’18* ACM Press, New York, USA, pp. 1–6.