A VARIANT SYSTEM FOR PATIENT MONITORING USING INTERNET OF THINGS

A PROJECT REPORT

Submitted by

M Kamal kumar	**(113313106032)**
Kambham Hema kumar	**(113313106033)**
S Kishore kumar	**(113313106038)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

VELAMMAL INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY::CHENNAI 600 025

APRIL 2017

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**A VARIANT SYSTEM FOR PATIENT MONITORING USING INTERNET OF THINGS** ” **is the bonafide work of** “M.Kamal kumar (113313106033), Kambham Hemakumar (113313106033), S Kishore kumar (113313106038)” **who carried out the project work under my supervision.**

SIGNATURE	**SIGNATURE**
Mr. S. Ilaiyaraja, M.E., (Ph.D)	**Mr. G.Sethuram Rao, M.E., (Ph.D)**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
	Assistant Professor
Electronics and Communication	Electronics and Communication
Velammal Institute of Technology	Velammal Institute of Technology
Panchetti, Chennai – 601 204	Panchetti, Chennai – 601 204

Submitted for the project Viva – Voce examination held on ______________________ at Velammal Institute of Technology.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

The satisfactions that accompany the successful completion of any task would be incomplete without mentioning the people who made it possible by their constant guidance and encouragement crowned our efforts with success.

We are grateful to our Chairman **Shri. M. V. Muthuramalingam,** for facilitating us with this opportunity. Our sincere thanks to our Director **Shri. M. V. M. Sasikumar**, Advisor's **Mr. K. Razak,** and **Mr. M. Vaasu,** Principal **Dr. T. Chandrashekar**, Vice Principal **Dr. S. Soundararajan,** for their support.

Our respected Head of the Department of Electronics and Communication Engineering **Mr. S. Ilaiyaraja**, Project Coordinator's **Mr. L. Balaji,** and **Dr. A. Benjamin Joseph,** Supervisor **Mr.G.Sethuram Rao,** deserve a special note of thanks and gratitude for having extended their fullest cooperation and continuous suggestions to make this project successful. We also thank all the staff members of Electronics and Communication Engineering department for their help during the course of this project.

We also like to express our gratefulness to our beloved parents and our family members who have always provided backup with their unending moral support and of course momentary help too which we believe are the driving forces for the completion of the project.

M.Kamal kumar

Kambham Hemakumar

S.Kishore kumar

ABSTRACT

e-health services can take advantage of technological achievements and advancements, one such technology that is currently in the roll is the Internet of Things. This technique offers scope for monitoring data using the variety of aids, wearable sensors, mobile-applications, etc. Due to heterogeneity of connecting devices and vast human life patterns in the IoT environment, life logging personal information consists of huge uncertainty and hardly used for health care studies. It is for this reason A variant system is proposed where the foremost goal is to design and demonstrate an innovative web based remote health care diagnostic system. One of the key challenges India is facing today is: Rural versus Urban divide in Healthcare. World Health Organization ranks India's healthcare system at 112 out of 190 countries. For those living in urban areas, high quality healthcare is readily available. But a staggering 70% of our population lives in rural India and don't get the same healthcare facilities as their urban counterpart. In contrast urban centers have specialist doctors and high quality clinics to provide quality healthcare. In this scenario, how the country should transform its healthcare system is the biggest question that needs to be answered.

This project aims to design and demonstrate an innovative web based remote healthcare diagnostic system that provides vital medical data and live video images of a patient situated in rural area accessible to a health professional available elsewhere in urban centers resulting in better diagnosis and treatment of the patient.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBERIVIATIONS

ADC	Analog to digital converter
CPU	Central Processing Unit
DAC	Digital to analog convertor
FMC	Flash Memory Chips
GPIO	General purpose input output
I2C	Inter integrated circuit
I2S	Inter integrated sound
IoT	Internet of Things
MIPS	Million Instructions Per Second
MP3	MPEG Layer-3 Sound File
MPEG	Motion Picture Experts Group
ROM	Read-Only Memory
RTOS	Real-Time Operating Systems
USART	Universal synchronous/asynchronous receiver transmitters

CHAPTER 1
INTRODUCTION

1.1 **ISSUES WITH EXISTING SYSTEMS**

Currently there are individual devices to monitor various health parameters a person. For remote monitoring, an individual caregiver is needed for monitoring and taking care of the patients. A person performs daily activities at regular interval of time. A growing amount of research id reported in recent times on development of a system to monitor the body vitals. At present, the health care units are using a lot of paperwork for maintaining patient's records regarding patient health condition. But if the records on paper are lost they are impossible to retrieve from the beginning.

1.2 **OVERVIEW OF PROPOSED SYSTEM**

The proposed system is designed to monitor the status of health parameters. Proposed system allows measuring various body vitals like heart rate, body temperature, body position, 3-lead ECG signal for continuous diagnostic purposes and remotely providing appropriate treatment plans. The system also defines a method to detect the QRS complex as a first step towards IoT Analytics, also a website is designed to publish the medical reports of the concerned patients. The advantages of the proposed system are low cost, easy access, simple design, and importantly a leap towards the IoT Analytics and web technology.

1.3 THE REVIEW ON INTERNET OF THINGS

The Internet of Things is an emerging topic of technical, social, and economic significance. Consumer products, durable goods, cars and trucks, industrial and utility components, sensors, and other everyday objects are being combined with Internet connectivity and powerful data analytic capabilities that promise to transform the way we work, live, and play. Projections for the impact of IoT on the Internet and economy are impressive, with some anticipating as many as 100 billion connected IoT devices and a global economic impact of more than $11 trillion by 2025.

At the same time, however, the Internet of Things raises significant challenges that could stand in the way of realizing its potential benefits. Attention-grabbing headlines about the hacking of Internet-connected devices, surveillance concerns, and privacy fears already have captured public attention. Technical challenges remain and new policy, legal and development challenges are emerging. The Internet of Things engages a broad set of ideas that are complex and intertwined from different perspectives.

Key concepts that serve as a foundation for exploring the opportunities and challenges of IoT include:

1.3.1 IoT Definition: The term Internet of Things generally refers to scenarios where network connectivity and computing capability extends to objects, sensors and everyday items not normally considered computers, allowing these devices to generate, exchange and consume data with minimal human intervention. There is, however, no single, universal definition.

1.3.2 Enabling Technologies: The concept of combining computers, sensors, and networks to monitor and control devices has existed for decades. The recent confluence of several technology market trends, however, is bringing the Internet of Things closer to widespread reality. These include *Ubiquitous Connectivity*, *Widespread Adoption of IP-based Networking*, *Computing Economics*, *Miniaturization*, *Advances in Data Analytics*, and the *Rise of Cloud Computing*.

1.3.3 Connectivity Models: IoT implementations use different technical communications models, each with its own characteristics. Four common communications models described by the Internet Architecture Board include: *Device-to-Device*, *Device-to-Cloud*, *Device-to-Gateway*, and *Back-End Data-Sharing*. These models highlight the flexibility in the ways that IoT devices can connect and provide value to the user.

1.3.4 Transformational Potential: If the projections and trends towards IoT become reality, it may force a shift in thinking about the implications and issues in a world where the most common interaction with the Internet comes from passive engagement with connected objects rather than active engagement with content. The potential realization of this outcome – a "hyper connected world" – is testament to the general-purpose nature of the Internet architecture itself, which does not place inherent limitations on the applications or services that can make use of the technology. Five key IoT issue areas are examined to explore some of the most pressing challenges and questions related to the technology. These include security; privacy; interoperability and standards; legal, regulatory, and rights; and emerging economies and development.

1.3.5 Security: While security considerations are not new in the context of information technology, the attributes of many IoT implementations present new and unique security challenges. Addressing these challenges and ensuring security in IoT products and services must be a fundamental priority. Users need to trust that IoT devices and related data services are secure from vulnerabilities, especially as this technology become more pervasive and integrated into our daily lives. Poorly secured IoT devices and services can serve as potential entry points for cyber attack and expose user data to theft by leaving data streams inadequately protected. The interconnected nature of IoT devices means that every poorly secured device that is connected online potentially affects the security and resilience of the Internet globally. This challenge is amplified by other considerations like the mass-scale deployment of homogenous IoT devices, the ability of some devices to automatically connect to other devices, and the likelihood of fielding these devices in unsecure environments.

As a matter of principle, developers and users of IoT devices and systems have a collective obligation to ensure they do not expose users and the Internet itself to potential harm. Accordingly, a collaborative approach to security will be needed to develop effective and appropriate solutions to IoT security challenges that are well suited to the scale and complexity of the issues.

1.3.6 Privacy: The full potential of the Internet of Things depends on strategies that respect individual privacy choices across a broad spectrum of expectations. The data streams and user specificity afforded by IoT devices can unlock incredible and unique value to IoT users, but concerns about privacy and potential harms might hold back full adoption of the Internet of Things. This means that privacy rights and respect for user privacy expectations are integral to ensuring user trust and confidence in the Internet, connected devices, and related services. Indeed, the Internet of Things is redefining the debate about privacy issues, as many implementations can dramatically change the ways personal data is collected, analyzed, used, and protected. For example, IoT amplifies concerns about the potential for increased surveillance and tracking, difficulty in being able to opt out of certain data collection, and the strength of aggregating IoT data streams to paint detailed digital portraits of users. While these are important challenges, they are not insurmountable. In order to realize the opportunities, strategies will need to be developed to respect individual privacy choices across a broad spectrum of expectations, while still fostering innovation in new technology and services.

1.3.7 Interoperability / Standards: A fragmented environment of proprietary IoT technical implementations will inhibit value for users and industry. While full interoperability across products and services is not always feasible or necessary, purchasers may be hesitant to buy IoT products and services if there is integration inflexibility, high ownership complexity, and concern over vendor lock-in. In addition, poorly designed and configured IoT devices may have negative consequences for the networking resources they connect to and the broader Internet. Appropriate standards, reference models, and best practices also will help curb the proliferation of devices that may act in disrupted ways to the Internet. The use of generic, open, and widely available standards as technical building blocks for IoT devices and services (such as the Internet Protocol) will support greater user benefits, innovation, and economic opportunity.

1.3.8 Legal, Regulatory and Rights: The use of IoT devices raises many new regulatory and legal questions as well as amplifies existing legal issues around the Internet. The questions are wide in scope, and the rapid rate of change in IoT technology frequently outpaces the ability of the associated policy, legal, and regulatory structures to adapt. One set of issues surrounds crossborder data flows, which occur when IoT devices collect data about people in one jurisdiction and transmit it to another jurisdiction with different data protection laws for processing. Further, data collected by IoT devices is sometimes susceptible to misuse, potentially causing discriminatory outcomes for some users. Other legal issues with IoT devices include the conflict between law enforcement surveillance and civil rights; data retention and destruction policies; and legal liability for unintended uses, security breaches or privacy lapses. While the legal and regulatory challenges are broad and complex in scope, adopting the guiding Internet Society principles of promoting a user's ability to *connect, speak, innovate, share, choose,* and *trust* are core considerations for evolving IoT laws and regulations that enable user rights.

1.3.9 Emerging Economy and Development Issues: The Internet of Things holds significant promise for delivering social and economic benefits to emerging and developing economies. This includes areas such as sustainable agriculture, water quality and use, healthcare, industrialization, and environmental management, among others. As such, IoT holds promise as a tool in achieving the United Nations Sustainable Development Goals. The broad scope of IoT challenges will not be unique to industrialized countries. Developing regions also will need to respond to realize the potential benefits of IoT. In addition, the unique needs and challenges of implementation in less-developed regions will need to be addressed, including infrastructure readiness, market and investment incentives, technical skill requirements, and policy resources.

The Internet of Things is happening now. It promises to offer a revolutionary, fully connected "smart" world as the relationships between objects, their environment, and people become more tightly intertwined. Yet the issues and challenges associated with IoT need to be considered and addressed in order for the potential benefits for individuals, society, and the economy to be realized.

Ultimately, solutions for maximizing the benefits of the Internet of Things while minimizing the risks will not be found by engaging in a polarized debate that pits the promises of IoT against its possible perils. Rather, it will take informed engagement, dialogue, and collaboration across a range of stakeholders to plot the most effective ways forward.

1.4 EMBEDDED SYSTEM

An Embedded system is a computer system designed for specific control functions within a larger system, often with real time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a Personal Computer (PC), is designed to be flexible and to meet a wide range of end-user needs.

Embedded systems contain processing cores that are either microcontrollers or Digital Signal Processors (DSP). A processor is an important unit in the embedded system hardware. It is the heart of the embedded system. The key characteristic, however, is being dedicated to handle a particular task, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Physically, embedded systems range from portable devices such as digital watches and MPEG Layer-3 Sound File (MP3) players, to large stationary installations like traffic lights and factory controllers. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

1.4.1 HISTORY OF EMBEDDED SYSTEMS

One of the first recognizably modern embedded systems was the Apollo Guidance Computer, developed by Charles Stark Draper at the Massachusetts Institute of Technology Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the Autonetics D-17 guidance computer for the Minuteman missile, released in 1961. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits. This program alone reduced prices on quad nand gate Integrated Circuits (IC) from $1000/each to $3/each, permitting their use in commercial products.

Since these early applications in the 1960s, embedded systems have come down in price and there has been a dramatic rise in processing power and functionality. The first microprocessor for example, the Intel 4004, was designed for calculators and other small systems but still required many external memory and support chips. In 1978 National Engineering Manufacturers Association released a "standard" for programmable microcontrollers, including almost any computer-based controllers, such as single board computers, numerical, and event-based controllers.

As the cost of microprocessors and microcontrollers fell it became feasible to replace expensive knob-based analog components such as potentiometer and variable capacitors with up/down buttons or knobs read out by a microprocessor even in some consumer products. By the mid-1980s, most of the common previously external system components had been integrated into the same chip as the processor and this modern form of the microcontroller allowed an even more widespread use, which by the end of the decade were the norm rather than the exception for almost all electronics devices.

The integration of microcontrollers has further increased the applications for which embedded systems are used into areas where traditionally a computer would not have been considered. A general purpose and comparatively low-cost microcontroller may often be programmed to fulfill the same role as a large number of separate components. Although in this context an embedded system is usually more complex than a traditional solution, most of the complexity is contained within the microcontroller itself. Very few additional components may be needed and most of the design effort is in the software. The intangible nature of software makes it much easier to prototype and test new revisions compared with the design and construction of a new circuit not using an embedded processor.

1.4.2 CHARACTERISTICS OF EMBEDDED SYSTEMS

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

Embedded systems are not always standalone devices. Many embedded systems consist of small, computerized parts within a larger device that serves a more general purpose. The program instructions written for embedded systems are referred to as firmware, and are stored in Read-Only Memory (ROM) or Flash Memory Chips (FMC). They run with limited computer hardware resources: little memory, small or non-existent keyboard or screen.

1.4.2.1 Reliability and Predictability

Embedded systems must be reliable. Depending on the application, the system might need to operate for long periods without human intervention. Embedded systems are also real-time systems, meeting time requirements is key to ensuring proper operation. The term deterministic or predictability describes Real-Time

Operating Systems (RTOS) with predictable behavior, in which the completion of operating system calls occurs within known timeframes.

1.4.2.2 Performance

This requirement dictates that an embedded system must perform fast enough to fulfill its timing requirements. Typically, the more deadlines to be met-and the shorter the time between them-the faster the system's Central Processing Unit (CPU) must be. The processor's performance is expressed in Million Instructions Per Second (MIPS).

1.4.2.3 Compactness

Application design constraints and cost constraints help determine how compact an embedded system can be. These design requirements limit system memory, which in turn limits the size of the application and operating system**.**

1.4.2.4 Scalability

Embedded system must be able to scale up or down to meet application specific requirements. Depending on how much functionality is required, it should be capable of adding or deleting modular components, including file systems and protocol stacks.

1.4.3 EMBEDDED SYSTEM DESIGN

We have to implement two parts in an embedded system. It’s depending upon the application requirements of the embedded project.

1. Hardware for speed and performance.
2. Software for flexibility**.**

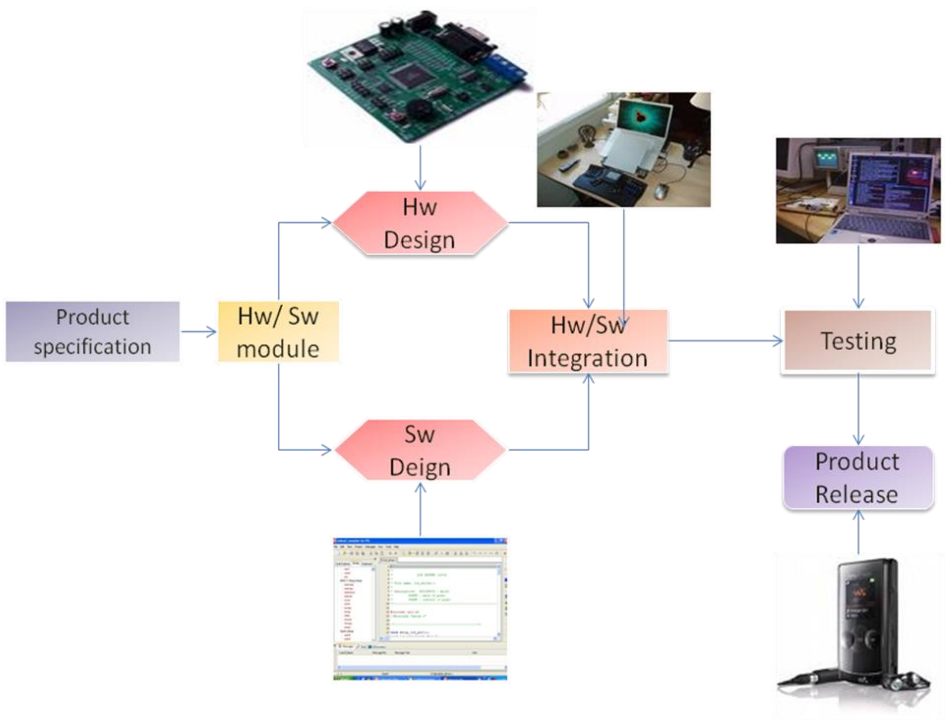


Fig 1 Embedded system design module

1.4.3.1 Hardware Design

Hardware design it gives physical appearance to the embedded system depending upon the application requirements. It's like microcontroller, peripherals, timer, memory, external panel for physical look and application required components for designing.

1.4.3.2 Software module design

- Systems are complex
- Classical design methodologies
    - informal design
    - formal final code
    - verification of the code by testing
- Bugs often revealed late while testing
- Several bugs remain undetected

- Heavy redesign cost

1.4.4 EMBEDDED TOOLS

1.4.4.1 Compiler

A software-development tool that translates high-level language programs into the machine-language instructions that a particular processor can understand and execute.

1.4.4.2 Linker

A software development tool that accepts one or more object files as input and outputs a re-locatable program or executable program.

1.4.4.3 Debugger

A debugger is a computer program that is used to test and debug other programs. The code to be examined might alternatively be running on an instruction set.

1.4.4.4 In-Circuit Emulator

An In-Circuit Emulator (ICE) is a hardware device used to debug the software of an embedded system. It is usually in the form of bond-out processor which has many internal signals brought out for the purpose of debugging on the target board. Emulator used to Load programs into the embedded system. To run, to see and change the data used by the system's software.

1.4.5 ADVANTAGE OF EMBEDDED SYSTEM

- They are easy to use and maintain.
- Customization yields lower area, power and cost.

1.4.6 EMBEDDED SYSTEM APPLICATION

- Consumer products, e.g., washers, microwave ovens, ...
- Automobiles, e.g., anti-lock braking, engine control, ...
- Industrial process controllers & defense applications.
- Computer/communication products, e.g., printers, Facsimile (FAX), ...

- Emerging multimedia applications & consumer electronics e.g., cell phones, ...

1.4.7 THE FUTURE OF EMBEDDED SYSTEM

In recent years, the embedded industry has experienced dramatic transformation, viz ,

- Product market windows now dictate feverish six- to nine-month turnaround cycles.
- Globalization is redefining market opportunities and expanding application space.
- Connectivity is now a requirement rather than a bonus in both wired and emerging wireless technologies.
- Interconnecting embedded systems are yielding new applications that are dependent on networking infrastructures.

Embedded software will continue to proliferate into new applications and lead to smarter classes of products. With an ever-expanding marketplace fortified by growing consumer demand for devices that can virtually run themselves as well as the seemingly limitless opportunities created by the Internet, embedded systems will continue to reshape the world for years to come.

CHAPTER 2
LITERATURE REVIEW

TITLE- Remote health monitoring system for detecting cardiac disorders
Author-Ayush Bansal, Sunil Kumar, Anurag Bajpai, Vijay N. Tiwari, Mithun Nayak, Shankar Venkatesan, Rangavittal Narayanan (2015)
Challenges-Remote health monitoring system with clinical decision support system as a key component could potentially quicken the response of medical specialists

TITLE- Using off-the-shelf medical devices for biomedical signal monitoring in a telemedicine system for emergency medical services
Author-Sebastian Thelen, Michael Czaplik, Philipp Meisen, Daniel Schilberg, and Sabina Jeschke, Senior Member, IEEE (2013)

TITLE- Continuous ECG monitoring in the management of pre-hospital health emergencies
Author-F Chiarugi1 , D Trypakis , V Kontogiannis , PJ Lees , CE Chronaki , M Zeaki , N Giannakoudakis , D Vourvahakis , M Tsiknakis , SC Orphanoudakis, IEEE (2013)

TITLE-A smart health monitoring chair for nonintrusive measurement of biological signals
Author-Hyun Jae Back, Gih Sung Chung, KoKeun Kim, Kwang Suk Park (2012)
Challenges-Used to predict values but it does not transmit information through any medium

CHAPTER 3
PROPOSED METHODOLOGY

3.1 INTERNET BASED PATIENT MONITORING AND DIAGNOSIS SYSTEM:

This project aims to design and demonstrate an innovative web based remote healthcare diagnostic system that provides vital medical data and live video images of a patient situated in rural area accessible to a health professional available elsewhere in urban centers resulting in better diagnosis and treatment of that patient.

Various medical sensors are used to collect patient medical data and sent to the monitoring station for interpretation.

1) 3-channel ECG
2) Blood pressure
3) Respiration rate
4) Body temperature
5) Body position
6) The system does include an advanced camera sensor to provide live video images of the patient.

Embedded Web Server

The project uses an ARM Cortex-M4 based microcontroller that acts as a web server that sends data to the client side application. Any device with an Internet browser such as Smartphone or PC/Laptop can be used to monitor the data feed. The client side user is authenticated with a unique user name and password before accessing streaming content. The web server is responsible for serving the web pages, servicing the client request and for maintaining the TCP/IP connection until the user ends the session. Web pages are constructed with HTML language. The device uses the LwIP open source TCP/IP protocol stack for its internet connectivity.

Medical Sensors

A 3-channel analog signal conditioning circuitry based on AD8232 is used to measure ECG signal. ECG represents the electrical activity of the heart and helps to diagnose various heart conditions. The circuitry is designed to extract, amplify, and filter the very small bio-potential signals generated from the heart. Three pins RA (Right Arm), LA (Left Arm), and RL (Right Leg) are used to attach the sensor leads to the patient body. A typical ECG signal is shown below.

A digital blood pressure sensor is used to measure patient blood pressure. It consists of an inflatable cuff and an analog pressure sensor circuitry. Pulse rate is measured from the ECG signal. Respiration rate sensor calculates a patient’s breathing rate by detecting changes in temperature when the patient breathes in and out. A 3-axis MEMS accelerometer sensor measures the body position and an analog temperature sensor measures body temperature.

Live Video Feed

The system consists of a camera sensor to stream a live video feed. When request is made the onboard microcontroller captures the JPEG images from the camera using the built-in DCMI peripheral and starts to stream it over the web in MJPEG compression format at an acceptable rate between. The image resolution is fixed at 470 x 272. The microcontroller has a large RAM memory area, about 256KB, which is a must for this kind of application.

Security

Before connecting with the system the user has to enter the login username and password which is a much needed security measure to prevent others from accessing the content. Once logged in, the user allowed accessing all the data including the video feed.

Embedded RTOS

A real time operating system is necessary to handle the timely events and other multitasking requirements of the project. Here FreeRTOS is chosen to provide this ability. FreeRTOS is the number one real-time operating system in the world.

3.2 BLOCK DIAGRAM OF PROPOSED SYSTEM

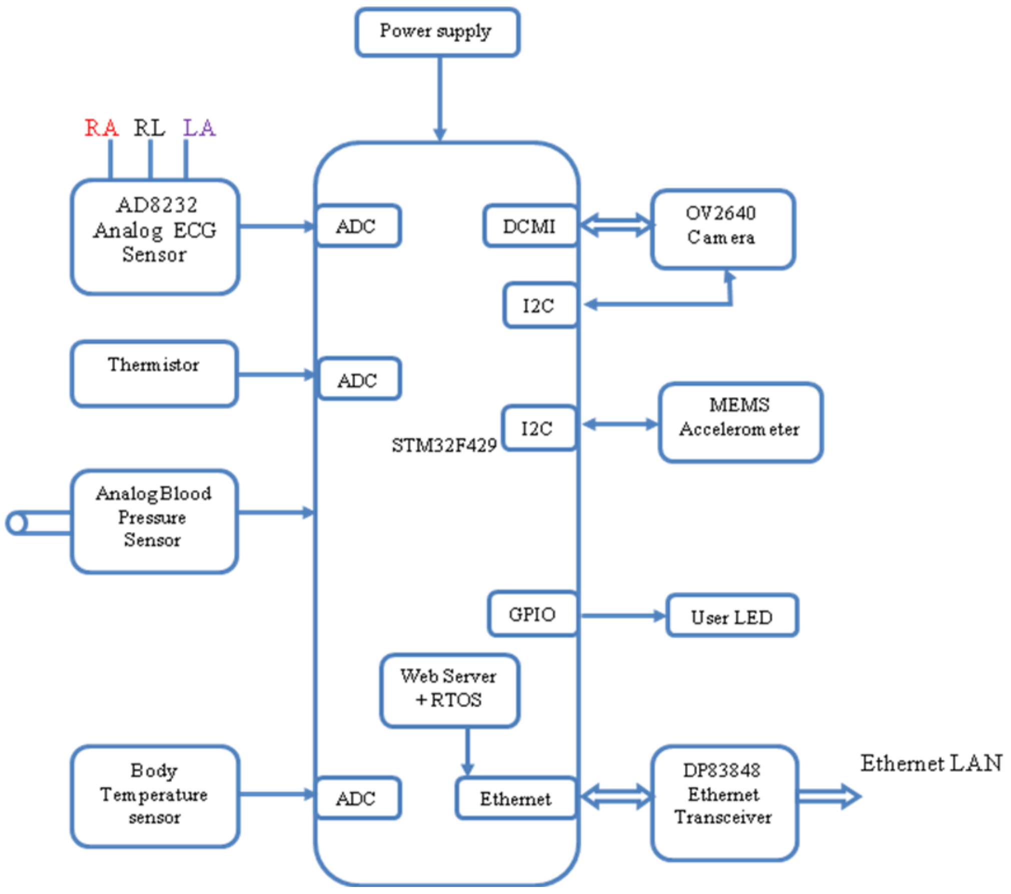


Figure 2 block diagram of proposed system

3.3 ARM Cortex-M4:

The **ARM Cortex™-M4** processor is one of the latest embedded processor by **ARM** specifically developed to address digital signal control markets that demand an efficient, easy-to-use blend of control and signal processing capabilities. The ARM® Cortex®-M4 processor is a high performance embedded processor with DSP instructions developed to address digital signal control markets that demand an efficient, easy-to-use blend of control and signal processing capabilities. The processor is highly configurable enabling a wide range of implementations from those requiring floating point operations, memory protection and powerful trace technology to cost sensitive devices requiring minimal area.

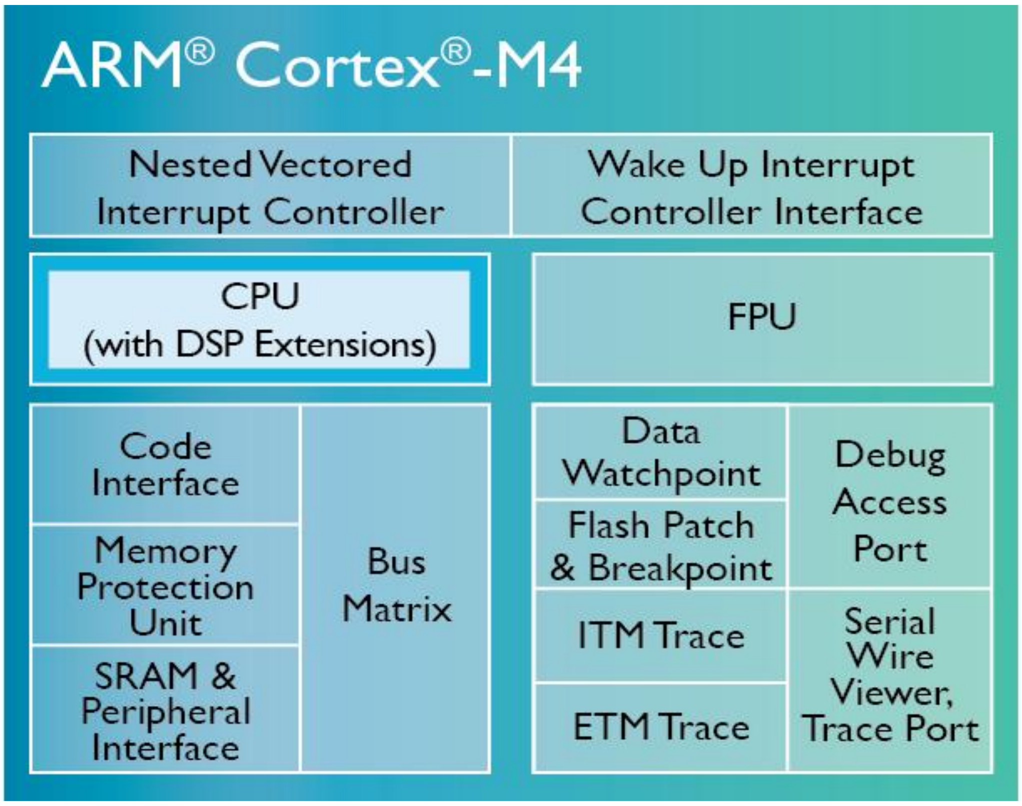

ARM® Cortex®-M4
Nested Vectored Interrupt Controller
Wake Up Interrupt Controller Interface
CPU (with DSP Extensions)
FPU
Code Interface
Memory Protection Unit
SRAM & Peripheral Interface
Bus Matrix
Data Watchpoint
Flash Patch & Breakpoint
ITM Trace
ETM Trace
Debug Access Port
Serial Wire Viewer, Trace Port

Figure 3 ARM Cortex –M4

The ARM® Cortex®-M4 processor is an award winning processor specifically developed to address digital signal control markets that demand an efficient, easy-to-use blend of control and signal processing capabilities. In the EFM32™ Wonder Gecko, the combination of high-efficiency signal processing functionality with the proven energy friendly Gecko technology makes for an easy-to-use microcontroller with the lowest energy consumption available.

3.3.1 Cortex-M4 with FPU and Signal Processing Technologies

The Cortex-M4 processor has been designed with a large variety of highly efficient signal processing features applicable to digital signal control markets. The processor features extended single-cycle multiply-accumulate (MAC) instructions, optimized SIMD arithmetic, saturating arithmetic instructions and an optional single precision Floating Point Unit (FPU). These features build upon the innovative technology that characterizes the ARM Cortex-M series processors.

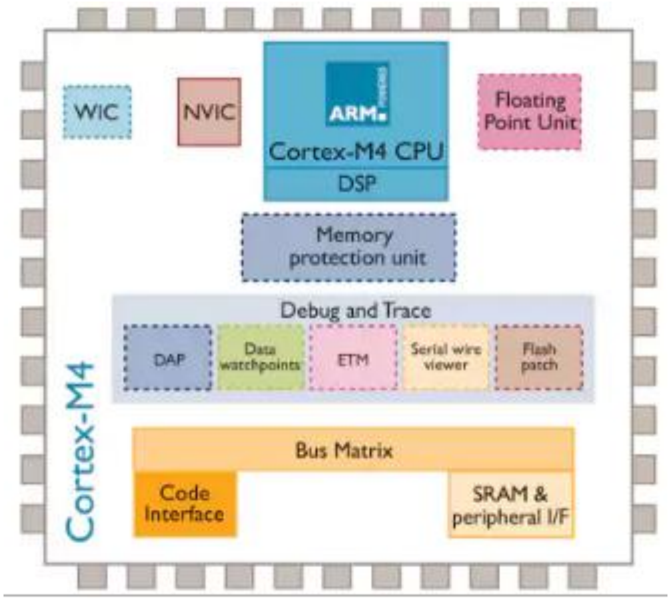


Figure 4 ARM Cortex M4 CPU DSP

3.3.2 Specifications

- IEEE 754 standard compliant
- Single precision floating point unit
- Fused MAC for higher precision

3.3.3 Key Benefits

- Gain the advantages of a microcontroller with integrated DSP, SIMD, and MAC instructions that simplify overall system design, software development and debug
- Accelerate single precision floating point math operations up to 10x over the equivalent integer software library with the optional floating point unit (FPU)
- Develop solutions for a large variety of markets with a full-featured ARMv7-M instruction set that has been proven across a broad set of embedded applications
- Achieve exceptional 32-bit performance with low dynamic power, delivering leading system energy efficiency due to integrated software controlled sleep modes, extensive clock gating and optional state retention.

3.3.4 Features

- High-efficiency signal processing
- Low-power
- Low-cost
- Easy to use

3.4 STM32F429:

The STM32F429/439 lines offer the performance of the Cortex-M4 core (with floating point unit) running at 180 MHz while reaching lower static power consumption (Stop mode) versus STM32F405/415/407/F417.

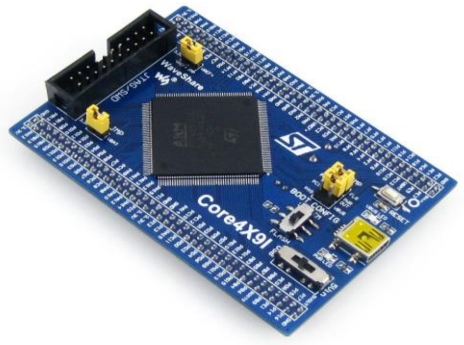


Figure 5 STM32F429 Microcontroller

3.4.1 Performance: At 180 MHz, the STM32F429/439 deliver 225 DMIPS/608 CoreMark performance executing from Flash memory, with 0-wait states thanks to ST's ART Accelerator. The DSP instructions and the floating point unit enlarge the range of addressable applications.

3.4.2 Power efficiency: ST's 90 nm process, ART Accelerator and the dynamic power scaling enables the current consumption in run mode and executing from Flash memory to be as low as 260 µA/MHz at 180 MHz In Stop mode, the power consumption is 120 µA typical, which is 3 times lower versus STM32F405/415/407/F417.

3.4.3Graphics: The new LCD-TFT controller interface with dual-layer support takes advantage of Chrom-ART Accelerator™. This graphics accelerator is performing content creation twice as fast as the core alone. As well as efficient 2-D raw data copy, additional functionalities are supported by the Chrom-ART Accelerator such as image format conversion or image blending (image mixing with some transparency). As a result, the Chrom-ART Accelerator boosts graphics content creation and saves processing bandwidth of the MCU core for the rest of the application.

3.4.4 Integration:

- Audio: 2 dedicated audio PLL, 2 full duplex I²S and a new serial audio interface (SAI) supporting time division multiplex (TDM) mode
- Up to 20 communication interfaces (including 4x USARTs plus 4x UARTs running at up to 11.25 Mbit/s, 6x SPI running at up to 45 Mbit/s, 3x I²C with a new optional digital filter capability, 2x CAN, SDIO)
- Analog: two 12-bit DACs, three 12-bit ADCs reaching 2.4 MSPS
- Up to 17 timers: 16- and 32-bit running at up to 180 MHz
- Easily extendable memory range using the flexible 90 MHz memory controller with 32-bit parallel interface, and supporting Compact Flash, SRAM, PSRAM, NOR, NAND and SDRAM memories
- Analog true random number generator
- The STM32F439 integrates a crypto/hash processor providing hardware acceleration for AES-128, -192 and -256, with support for GCM and CCM, Triple DES, and hash (MD5, SHA-1 and SHA-2)

The STM32F429 and STM32F439 portfolio provides from 512-Kbyte Flash to 2-Mbyte dual-bank Flash, 256-Kbyte SRAM and from 100 to 216 pins in packages as small as 5 x 5.1 mm. With such memory integration, the need for external memory is reduced, allowing smaller, safer and low-emission PCB designs.

The ART Accelerator™ is a memory accelerator which is optimized for STM32 industry-standard ARM® Cortex®-M4 with FPU processors. It balances the inherent performance advantage of the ARM® Cortex®-M4 with FPU over Flash memory technologies, which normally requires the processor to wait for the Flash memory at higher frequencies. To release the processor full 225 DMIPS performances at this frequency, the accelerator implements an instruction pre fetch queue and branch cache, which increases program execution speed from the 128-bit Flash memory. Based on CoreMark benchmark, the performance achieved thanks to the ART Accelerator is equivalent to 0 wait state program execution from Flash memory at a CPU frequency up to 180 MHz

3.4.5 Memory protection unit

The memory protection unit (MPU) is used to manage the CPU accesses to memory to prevent one task to accidentally corrupt the memory or resources used by any other active task. This memory area is organized into up to 8 protected areas that can in turn be divided up into 8 subareas. The protection area sizes are between 32 bytes and the whole 4 gigabytes of addressable memory. The MPU is especially helpful for applications where some critical or certified code has to be protected against the misbehavior of other tasks. It is usually managed by an RTOS (real- time operating system). If a program accesses a memory location that is prohibited by the MPU, the RTOS can detect it and take action. In an RTOS environment, the kernel can dynamically update the MPU area setting, based on the process to be executed.

3.4.6 Embedded Flash memory

The devices embed a Flash memory of up to 2 Mbytes available for storing programs and data.

3.4.7 Power supply schemes

• VDD = 1.7 to 3.6 V: external power supply for I/Os and the internal regulator (when enabled), provided externally through VDD pins.

• VSSA, VDDA = 1.7 to 3.6 V: external analog power supplies for ADC, DAC, Reset blocks, RCs and PLL. VDDA and VSSA must be connected to VDD and VSS, respectively.

• VBAT = 1.65 to 3.6 V: power supply for RTC, external clock 32 kHz oscillator and backup registers (through power switch) when VDD is not present

3.4.8 Inter-integrated circuit interface

Up to three I²C bus interfaces can operate in multi master and slave modes. They can support the standard (up to 100 KHz), and fast (up to 400 KHz) modes. They support the 7/10-bit addressing mode and the 7-bit dual addressing mode (as slave). A hardware CRC generation/verification is embedded. They can be served by DMA and they support SMBus 2.0/PMBus.

3.4.9 Universal synchronous/asynchronous receiver transmitters

The devices embed four universal synchronous/asynchronous receiver transmitters (USART1, USART2, USART3 and USART6) and four universal asynchronous receiver transmitters (UART4, UART5, UART7, and UART8). These six interfaces provide asynchronous communication, IrDA SIR ENDEC support, multiprocessor communication mode, single-wire half-duplex communication mode and have LIN Master/Slave capability. The USART1 and USART6 interfaces are able to communicate at speeds of up to 11.25 Mbit/s. The other available interfaces communicate at up to 5.62 bit/s. USART1, USART2, USART3 and USART6 also provide hardware management of the CTS and RTS signals, Smart Card mode (ISO 7816 compliant) and SPI-like communication capability. All interfaces can be served by the DMA controller.

3.4.10 Serial peripheral interface

The devices feature up to six SPIs in slave and master modes in full-duplex and simplex communication modes. SPI1, SPI4, SPI5, and SPI6 can communicate at up to 45 Mbits/s, SPI2 and SPI3 can communicate at up to 22.5 Mbit/s. The 3-bit pre scalar gives 8 master mode frequencies and the frame is configurable to 8 bits or 16 bits. The hardware CRC generation/verification supports basic SD Card/MMC modes. All SPIs can be served by the DMA controller. The SPI interface can be configured to operate in TI mode for communications in master mode and slave mode.

3.4.11 Inter-integrated sound

Two standard I2S interfaces (multiplexed with SPI2 and SPI3) are available. They can be operated in master or slave mode, in full duplex and simplex communication modes, and can be configured to operate with a 16-/32-bit resolution as an input or output channel. Audio sampling frequencies from 8 kHz up to 192 kHz are supported. When either or both of the I2S interfaces is/are configured in master mode, the master clock can be output to the external DAC/CODEC at 256 times the sampling frequency. All I2Sx can be served by the DMA controller.

3.4.12 Ethernet MAC interface with dedicated DMA and IEEE 1588 support

The devices provide an IEEE-802.3-2002-compliant media access controller (MAC) for Ethernet LAN communications through an industry-standard medium-independent interface (MII) or a reduced medium-independent interface (RMII). The microcontroller requires an external physical interface device (PHY) to connect to the physical LAN bus (twisted-pair, fiber, etc.). The PHY is connected to the device MII port using 17 signals for MII or 9 signals for RMII, and can be clocked using the 25 MHz (MII) from the microcontroller.

The devices include the following features:

• Supports 10 and 100 Mbit/s rates

• Dedicated DMA controller allowing high-speed transfers between the dedicated SRAM and the descriptors.

• Tagged MAC frame support (VLAN support)

• Half-duplex (CSMA/CD) and full-duplex operation

• MAC control sub layer (control frames) support

• 32-bit CRC generation and removal

• Several address filtering modes for physical and multicast address (multicast and group addresses)

• 32-bit status code for each transmitted or received frame

• Internal FIFOs to buffer transmit and receive frames. The transmit FIFO and the receive FIFO are both 2 Kbytes.

• Supports hardware PTP (precision time protocol) in accordance with IEEE 1588 2008 (PTP V2) with the time stamp comparator connected to the TIM2 input

• Triggers interrupt when system time becomes greater than target time

3.4.13 General-purpose input/outputs

Each of the GPIO pins can be configured by software as output (push-pull or open-drain, with or without pull-up or pull-down), as input (floating, with or without pull-up or pull-down) or as peripheral alternate function. Most of the GPIO pins are shared with digital or analog alternate functions. All GPIOs are high-current-capable and have speed selection to better manage internal noise, power consumption and electromagnetic emission. The I/O configuration can be locked if needed by following a specific sequence in order to avoid spurious writing to the I/Os registers. Fast I/O handling allowing maximum I/O toggling up to 90 MHz

3.4.14 Analog-to-digital converters (ADCs)

Three 12-bit analog-to-digital converters are embedded and each ADC shares up to 16 external channels, performing conversions in the single-shot or scan mode. In scan mode, automatic conversion is performed on a selected group of analog inputs.

Additional logic functions embedded in the ADC interface allow:

• Simultaneous sample and hold

• Interleaved sample and hold

The ADC can be served by the DMA controller. An analog watchdog feature allows very precise monitoring of the converted voltage of one, some or all selected channels. An interrupt is generated when the converted voltage is outside the programmed thresholds. To synchronize A/D conversion and timers, the ADCs could be triggered by any of TIM1, TIM2, TIM3, TIM4, TIM5, or TIM8 timer.

3.4.15 Digital-to-analog converter (DAC)

The two 12-bit buffered DAC channels can be used to convert two digital signals into two analog voltage signal outputs.

This dual digital Interface supports the following features:

• Two DAC converters: one for each output channel

- 8-bit or 10-bit monotonic output
- Left or right data alignment in 12-bit mode
- synchronized update capability
- Noise-wave generation
- Triangular-wave generation
- Dual DAC channel independent or simultaneous conversions
- DMA capability for each channel
- External triggers for conversion
- input voltage reference VREF+

Eight DAC trigger inputs are used in the device. The DAC channels are triggered through the timer update outputs that are also connected to different DMA streams.

3.5 OV2640 Camera Sensor

Figure 6 OV2640 Camera sensor

The OV2640 CAMERACHIP^{TM} is a low voltage CMOS image sensor that provides the full functionality of a single-chip UXGA (1632x1232) camera and image processor in a small footprint package. The OV2640 provides full-frame, sub-sampled, scaled or windowed 8-bit/10-bit images in a wide range of formats, controlled through the Serial Camera Control Bus (SCCB) interface.

This product has an image array capable of operating at up to 15 frames per second (fps) in UXGA resolution with complete user control over image quality, formatting and output data transfer. All required image processing functions, including exposure control, gamma, white balance, color saturation, hue control, white pixel canceling, noise canceling, and more, are also programmable through the SCCB interface. The OV2640 also includes a compression engine for increased processing power. In addition, OmniVision CAMERACHIPS use proprietary sensor technology to improve image quality by reducing or eliminating common lighting/electrical sources of image contamination, such as fixed pattern noise, smearing, etc., to produce a clean, fully stable color image.

3.5.1 Specification:

- JPEG output support
- Megapixel
- Supports image sizes: UXGA, SXGA, SVGA, and any size scaling down from SXGA to 40×30
- Standard SCCB and parallel camera interface connected to DCMI interface on STM32 and read using DMA
- High sensitivity for low-light operation
- Output support for Raw RGB, RGB (RGB565/555), GRB422, YUV (422/420) and YCbCr (4:2:2) formats
- Automatic image control functions
- Image quality control functions
- Maximum transfer rate UXGA/SXGA @15fps, UXGA/SXGA @30fps, SVGA @30fps, CIF @60fps
- Low operating voltage

3.5.2 Features:

- High sensitivity for low-light operation
- Low operating voltage for embedded portable apps
- Standard SCCB interface
- Output support for Raw RGB, RGB (RGB565/555), GRB422, YUV (422/420) and YCbCr (4:2:2) formats

- Supports image sizes: UXGA, SXGA, SVGA, and any size scaling down from SXGA to 40x30
- VarioPixel® method for sub-sampling
- Automatic image control functions including Automatic Exposure Control (AEC), Automatic Gain Control (AGC), Automatic White Balance (AWB), Automatic Band Filter (ABF), and Automatic Black-Level Calibration (ABLC)
- Image quality controls including color saturation, gamma, sharpness (edge enhancement), lens correction, white pixel canceling, noise canceling, and 50/60 Hz luminance detection.

3.5.3 Applications:

- Cellular and Camera Phones
- Toys
- PC Multimedia
- Digital Still Cameras

3.6 MEMS Accelerometer LIS302DL

Modern accelerometers are often small micro electro-mechanical systems (MEMS), and are indeed the simplest MEMS devices possible, consisting of little more than a cantilever beam with a proof mass (also known as seismic mass). Mechanically the accelerometer behaves as a mass-damper-spring system; the damping results from the residual gas sealed in the device. As long as the Q-factor is not too low, damping does not result in a lower sensitivity.

The LIS302DL is an ultra compact low-power three axes linear accelerometer. It includes a sensing element and an IC interface able to provide the measured acceleration to the external world through I2C/SPI serial interface. The sensing element, capable of detecting the acceleration, is manufactured using a dedicated process developed by ST to produce inertial sensors and actuators in silicon. The IC interface is manufactured using a CMOS process that allows designing a dedicated circuit which is trimmed to better match the sensing element characteristics. The LIS302DL has dynamically user selectable full scales of ±2g/±8g and it is capable of measuring accelerations with an output data rate of 100Hz or 400Hz. A self-test capability allows the user to check the functioning of the sensor in the final application. The device may be configured to generate inertial wake-up/free-fall interrupt signals when a programmable acceleration threshold is crossed at least in one of the three axes. Thresholds and timing of interrupt generators are completely programmable by the end user on the fly. The LIS302DL is available in plastic Thin Land Grid Array package (TLGA) and it is guaranteed to operate over an extended temperature range from -40°C to +85°C. The LIS302DL belongs to a family of products suitable for a variety of applications:

- Free-Fall detection
- Motion activated functions
- Gaming and Virtual Reality input devices
- Vibration Monitoring and Compensation

3.6.1 Feature:

- Three axes
- SPI/I2C digital interface
- Innovative embedded functionalities
- Two highly flexible and programmable interrupt request outputs
- Programmable thresholds and timing of interrupt signals
- 2.16V to 3.6V supply voltage
- 1.8V compatible I/Os
- <1mW power consumption
- Temperature range -40 to +85ºC
- ±2/±8g selectable full scale range

- Embedded high pass filter
- Embedded self-test
- 10,000g high shock survivability
- LGA package 3x5x0.9mm3
- ECOPACK® – RoHS Directive and green compliant

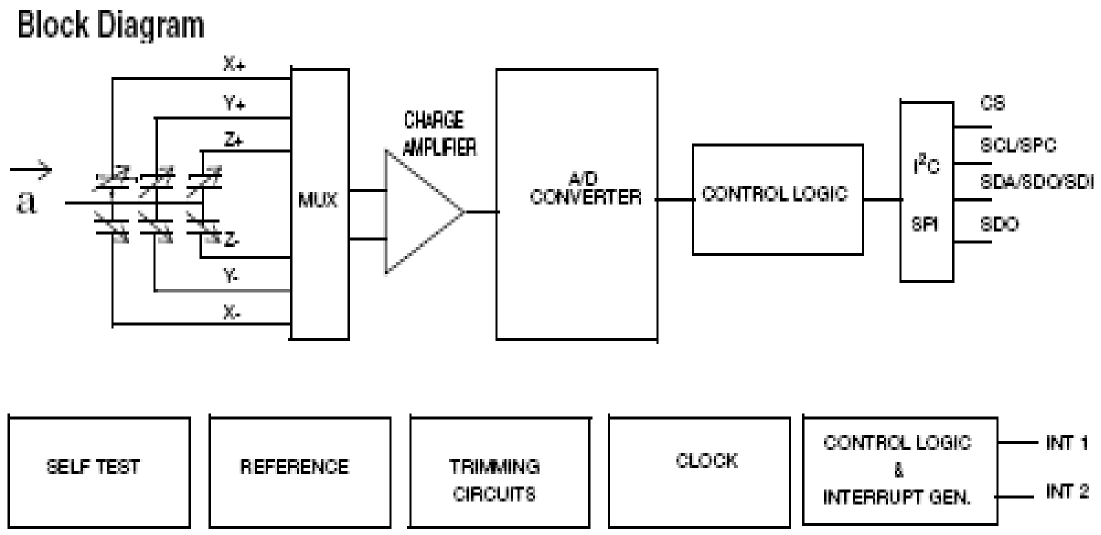


Figure 7 Block diagram of MEMS accelerometer sensor

Pin description LIS302DL

Pin#	**Name**	**Function**
1	Vdd_IO	Power supply for I/O pins
2	GND	0V supply
3	Reserved	Connect to Vdd
4	GND	0V supply
5	GND	0V supply
6	Vdd	Power supply
7	CS	SPI enable

		I2C/SPI mode selection (1: I2C mode; 0: SPI enabled)
8	INT 1	Inertial interrupt 1
9	INT 2	Inertial interrupt 2
10	GND	0V supply
11	Reserved	Connect to Gnd
12	SDO	SPI Serial Data Output
13	SDA	I2C Serial Data (SDA)
	SDI	SPI Serial Data Input (SDI)
	SDO	3-wire Interface Serial Data Output
14	SCL	I2C Serial Clock (SCL)
	SPC	SPI Serial Port Clock (SPC)

Table 1 PIN Description of MEMS Accelerometer LIS302DL

3.7 Temperature Sensor:

The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly-proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range.

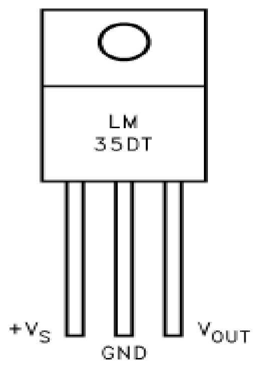


Figure 10 PIN Diagram for LM 35DT

Lower cost is assured by trimming and calibration at the wafer level. The low-output impedance, linear output and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 µA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy).

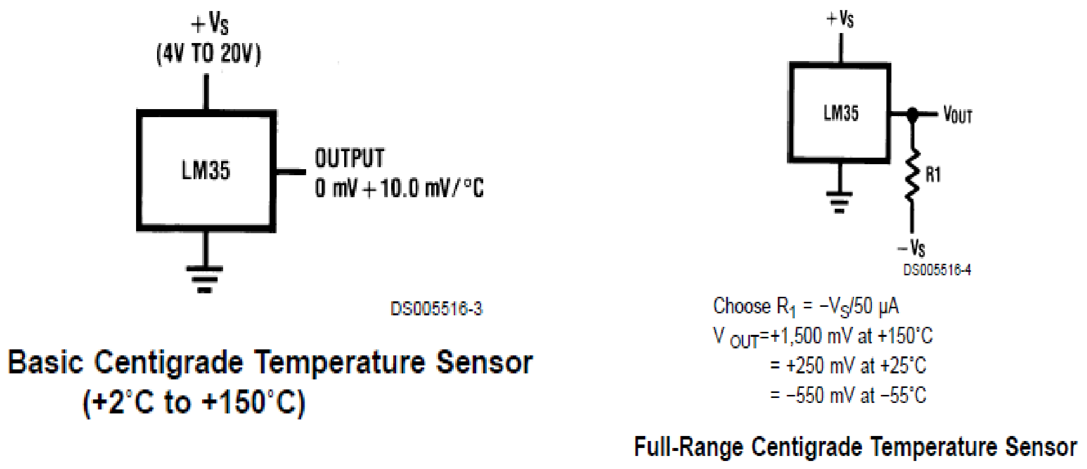


Figure 11 Range description of LM 35

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01˚C of the surface temperature. This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is especially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature. To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V− terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections. These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

3.7.1 Features:

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor

- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates from 4 V to 30 V
- Less than 60-µA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

3.8 Thermistor based Respiration Rate Sensor

Thermistor Sensor Module is used to sense temperature and convert it into output signals. It is associated with both analog output pin and digital output pin labeled as AO and DO respectively on the board. Thermistor Sensor Module is made of semiconductor materials. Basically Thermistor sensors are mostly Negative Temperature Coefficient (NTC), which means NTC Thermistor will have their electrical resistance decreased when subjected to an increase in body temperature. This module has a potentiometer knob that can be adjusted to change the sensitivity of Thermistor towards temperature.

Thermistor is a sensitive element and divided into two types, PTC and NTC, which we mentioned above, by different temperature coefficients. Its resistance varies significantly with temperature. The resistance of PTC thermistor increases with higher temperature when that of NTC, decreases. The precision of NTC Thermistor can be up to 0.1℃ and the time constant can be less than 10s. They are applied to thermo detectors in barn, as well as temperature measurement in food storing, medicine and health, ocean, etc.
the Thermistor sensor module measures temperature sensitively though it can output only analog signals.

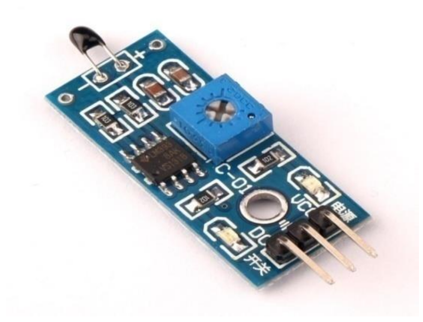


Figure 12 Thermistor based respiration rate measurement

The principle is that the resistance of the NTC Thermistor changes with the temperature difference in the outer environment. It detects the real-time temperature of the environment. When the temperature gets higher, the resistance of the Thermistor decreases and the voltage of pin A0 increases accordingly. The voltage data then is converted to digital quantities by the A/D adapter. The temperature in Celsius and Fahrenheit then is output via programming and then displayed on LCD1602.

3.8.1 Specification:

- Input Voltage: DC 3.3V to 5V
- Output: Analog and Digital
- Sensitivity Adjustable
- Size: 32mm x 14mm x 5mm

3.9 Analog ECG Sensor:

The AD8232 SparkFun Single Lead Heart Rate Monitor is a cost-effective board used to measure the electrical activity of the heart. This electrical activity can be charted as an ECG or Electrocardiogram and output as an analog reading. ECGs can be extremely noisy, the AD8232 Single Lead Heart Rate Monitor acts as an op amp to help obtain a clear signal from the PR and QT Intervals easily.

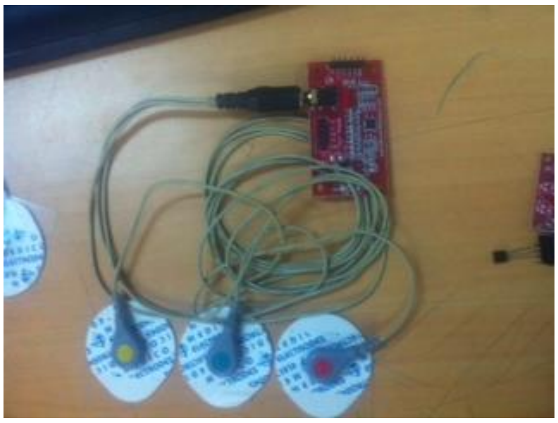


Figure 13 Analog ECG sensors AD 8232

The AD8232 is an integrated signal conditioning block for ECG and other bio potential measurement applications. It is designed to extract, amplify, and filter small bio potential signals in the presence of noisy conditions, such as those created by motion or remote electrode placement.

The AD8232 Heart Rate Monitor breaks out nine connections from the IC that you can solder pins, wires, or other connectors to. SDN, LO+, LO-, OUTPUT, 3.3V, GND provide essential pins for operating this monitor with an Arduino or other development board. Also provided on this board are RA (Right Arm), LA (Left Arm), and RL (Right Leg) pins to attach and use your own custom sensors. Additionally, there is an LED indicator light that will pulsate to the rhythm of a heart beat.

3.9.1 Features:

- Operating Voltage - 3.3V
- Analog Output
- Leads-Off Detection
- Shutdown Pin
- LED Indicator
- 3.5mm Jack for Biomedical Pad Connection

3.10 Analog Blood Pressure Sensor

The Barometric Pressure Sensor Module provides extremely accurate atmospheric pressure readings along with a reference temperature reading. The resolution allows altitude changes as small as 13cm to be detected.
The module includes an onboard voltage regulator and logic level shifters, allowing it to operate between 2.3V and 6V.

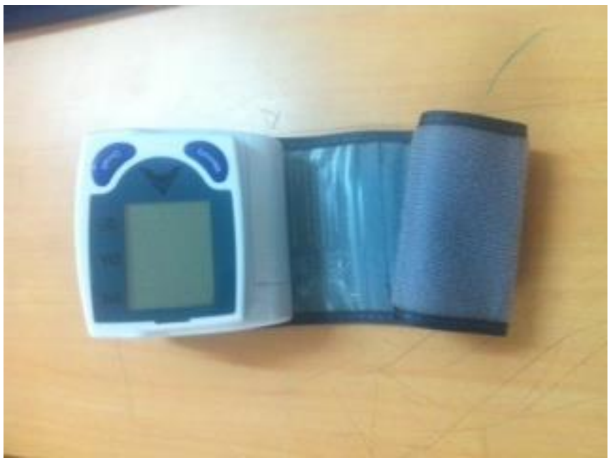


Figure 14 Analog Blood pressure sensors

3.10.1 Features

- Pressure and temperature sensor combined
- I2C interface for easy Arduino connection
- 2.3V to 6V operation using built-in level shifters and voltage regulator
- Onboard I2C pull-ups with optional isolation
- Maximum pressure sensor range: 10 to 2000 mbar (0.145 to 29 psi)
- Pressure sensor error band at 25C, within 300 to 1200mbar: +/-2mbar
- Pressure sensor resolution: 0.11 / 0.062 / 0.039 / 0.028 / 0.021 / 0.016 mbar
- Pressure sensor response time: 0.5 / 1.1 / 2.1 / 4.1 / 8.22 / 16.44 ms
- Pressure sensor stability: +/-1mbar / year
- Temperature sensor range: -40 to +85C
- Temperature sensor resolution: <0.01C
- Temperature sensor accuracy at 25C: +/-1C

CHAPTER 4
SOFTWARE TOOLS

4.1 Introduction of Atollic TrueSTUDIO IDE:

The Atollic TrueSTUDIO® IDE provides a modern and highly integrated development environment which directly supports the use of advanced workflow tools such as version control, bug tracking, code review, code analysis and distributed task-based development, along with tailored control for project and build control and a fully integrated debugger. The Atollic TrueSTUDIO® IDE comes in a variety of packages enabling customers to select the features/price model best suited to their development needs. As the underlying compiler tool chain is based on the GNU C/C++ compiler, there is no worry about a 'proprietary' toolchain becoming out of date, or unavailable. The same goes for the Atollic TrueSTUDIO® IDE, as it is based on the open Eclipse framework.
A standard firmware library infrastructure has been created by ARM Ltd. along with semiconductor and tool chain vendors. The Cortex® Microcontroller Software Interface Standard (CMSIS) defines a hardware abstraction layer which is available as a firmware library coded to support compilation by a number of compilers, including the GNU C/C++ compiler and the IAR EmbeddedWorkbench® C/C++ compiler. Details can be found on the ARM®.
The firmware generated by the Atollic TrueSTUDIO® IDE for the ARM® Cortex® series of processors includes all low-level device control via the CMSIS firmware library (including startup, interrupt and exception handlers) along with chip vendor supplied peripheral device drivers. As the firmware library complies to a standard, and has been written to support both the GNU and IAR EmbeddedWorkbench® compilers (by using conditional compilation), users should find that they have a familiar Application Programming Interface (API) to

code against, which reduces the porting exercise to one of tuning the build control and porting application source files.

4.2 CREATING AND SETTING UP A PROJECT

Atollic TrueSTUDIO® supports both managed and unmanaged projects. Managed projects are completely handled by the IDE and can be configured using GUI settings, whereas unmanaged projects require a makefile that has to be maintained manually.

Step 1: Select the File, New, C Project menu command to start the Atollic TrueSTUDIO® project wizard.

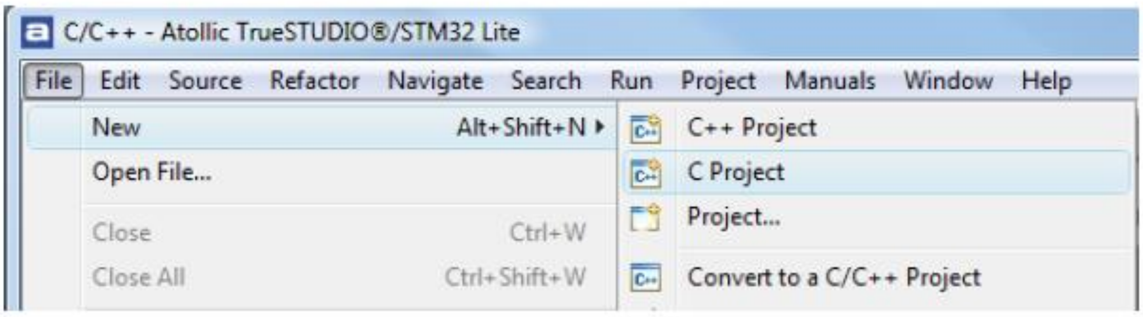
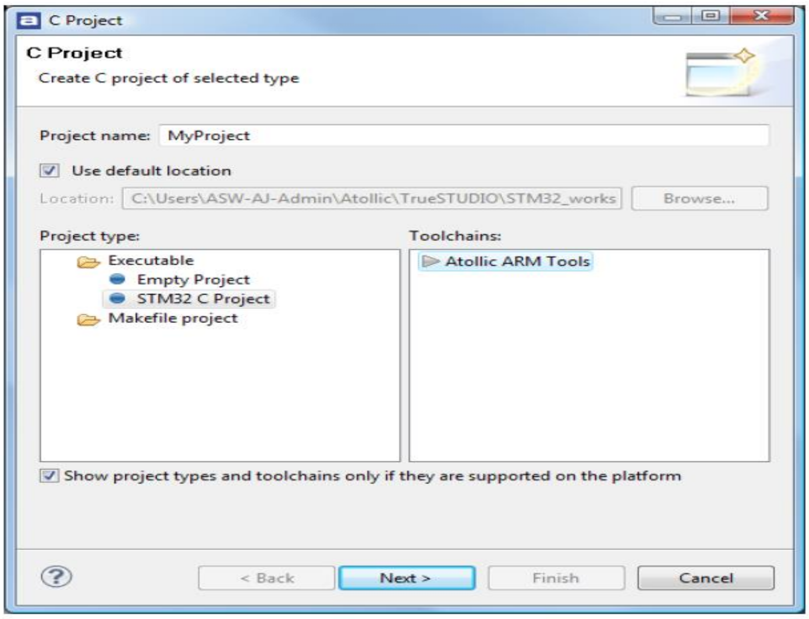


Figure 14 Atollic TrueSTUDIO® project wizard

Step 2: The C Project configuration page is displayed.



C Project

C Project

Create C project of selected type

Project name: MyProject

[✓] Use default location

Location: C:\Users\ASW-AJ-Admin\Atollic\TrueSTUDIO\STM32_works

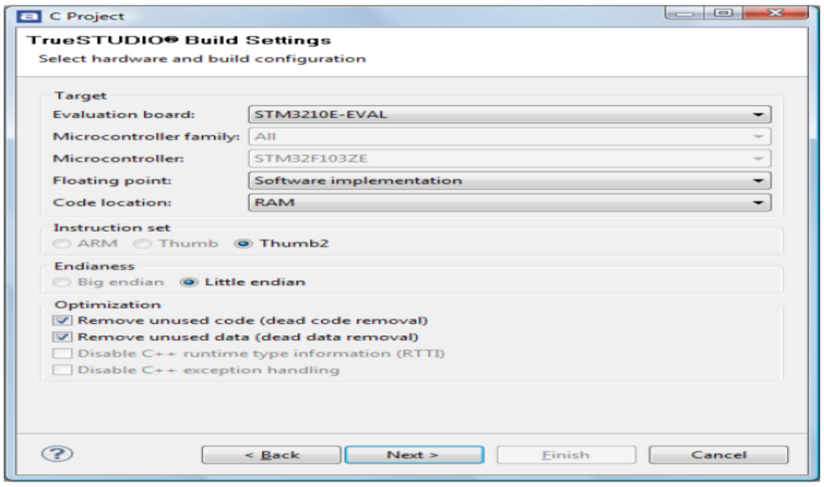
Project type:
- Executable
  - Empty Project
  - STM32 C Project
- Makefile project

Toolchains:
- Atollic ARM Tools

[✓] Show project types and toolchains only if they are supported on the platform

Figure 15 the C Project configuration page

Step 3: Enter a Project name (such as “My Project”) and select your embedded target as the Project type, and Atollic ARM Tools as the Tool chain. Then click the Next button to display the TrueSTUDIO® Build Settings page.



C Project

TrueSTUDIO® Build Settings

Select hardware and build configuration

Target

Evaluation board: STM3210E-EVAL

Microcontroller family: All

Microcontroller: STM32F103ZE

Floating point: Software implementation

Code location: RAM

Instruction set

ARM Thumb [✓ Thumb2]

Endianess

Big endian [✓ Little endian]

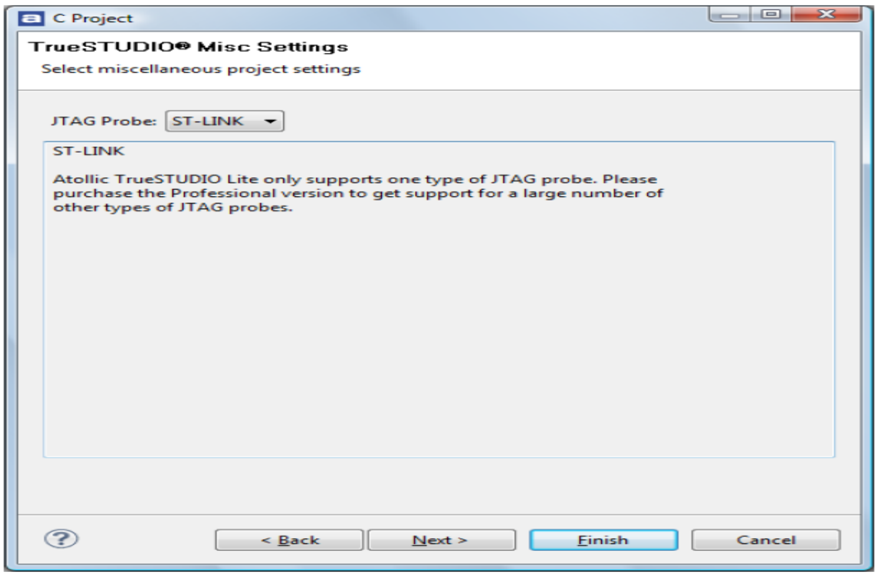
Optimization

- [✓] Remove unused code (dead code removal)
- [✓] Remove unused data (dead data removal)
- [ ] Disable C++ runtime type information (RTTI)
- [ ] Disable C++ exception handling

< Back | Next > | Finish | Cancel

Figure 16 the TrueSTUDIO® Build Settings page

Step 4: In the TrueSTUDIO® Build Settings page, configure the hardware settings according to your Evaluation board or custom board design. Finally, click the Next button to display the TrueSTUDIO® Debug Settings page.



C Project

TrueSTUDIO® Misc Settings

Select miscellaneous project settings

JTAG Probe: ST-LINK

ST-LINK

Atollic TrueSTUDIO Lite only supports one type of JTAG probe. Please purchase the Professional version to get support for a large number of other types of JTAG probes.

< Back | Next > | Finish | Cancel

Figure 17 the TrueSTUDIO® Debug Settings page

Step 5: Select the JTAG probe you use. Click the Next button to display the Select Configurations page.

Step 6: By default, Atollic TrueSTUDIO® builds the project automatically whenever any file in the build dependency is updated. This feature can be toggled using the Project, Build Automatically menu command.

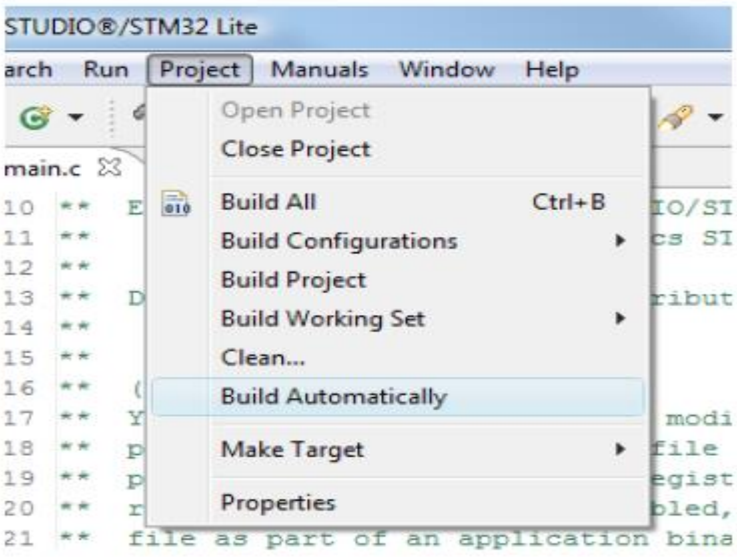
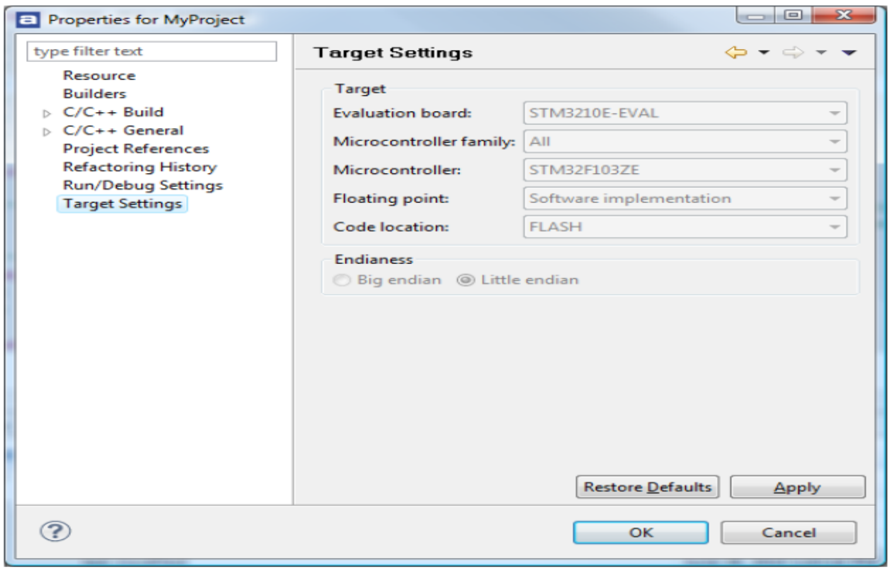


Figure 18 Build Automatically menu command

Step 7: Atollic TrueSTUDIO includes a very powerful graphical debugger based on the **gdb** command line debugger. When the configuration is completed, click the OK button to accept the new settings.



Target Settings

Target

Evaluation board: STM3210E-EVAL

Microcontroller family: All

Microcontroller: STM32F103ZE

Floating point: Software implementation

Code location: FLASH

Endianess

Big endian [✓ Little endian]

Figure 19 Atollic TrueSTUDIO gdb command line debugger

CHAPTER 5
RESULTS AND DISCUSSION

The data is collected from the different bio sensors are transmitted into the internet and the sample analysis of ECG is performed to detect the QRS complex using matlab. The received data is mostly analog in nature for demonstration purpose and feasibility, the data thus fetched is linked to a remote website and continuously updating the analysis report in that website manually which increases understanding the problems and their solutions regarding the medical domain.

As the number of parameters is measured, the validity of data can be compared with existing medical analysis reports that are available on the internet. This system facilitates continuous measurement of body vitals for monitory purposes. Also the system is developed with simplicity and easy access as priorities since it is mostly directed to be used at areas that are in need of remote access to medical checkups for a prolonged interval of time and in some critical diagnostic patients. The first step towards IoT Analytics is taken by demonstrating the analysis report and the real time data on the common web platform for simplified access.

CHAPTER 6

APPENDIX

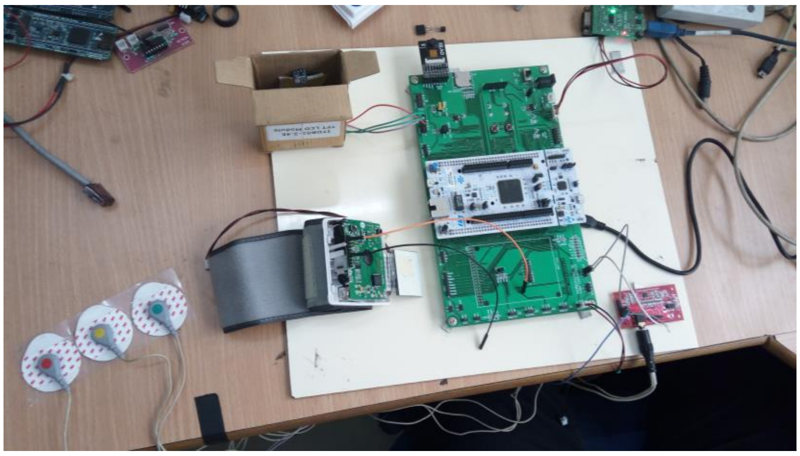
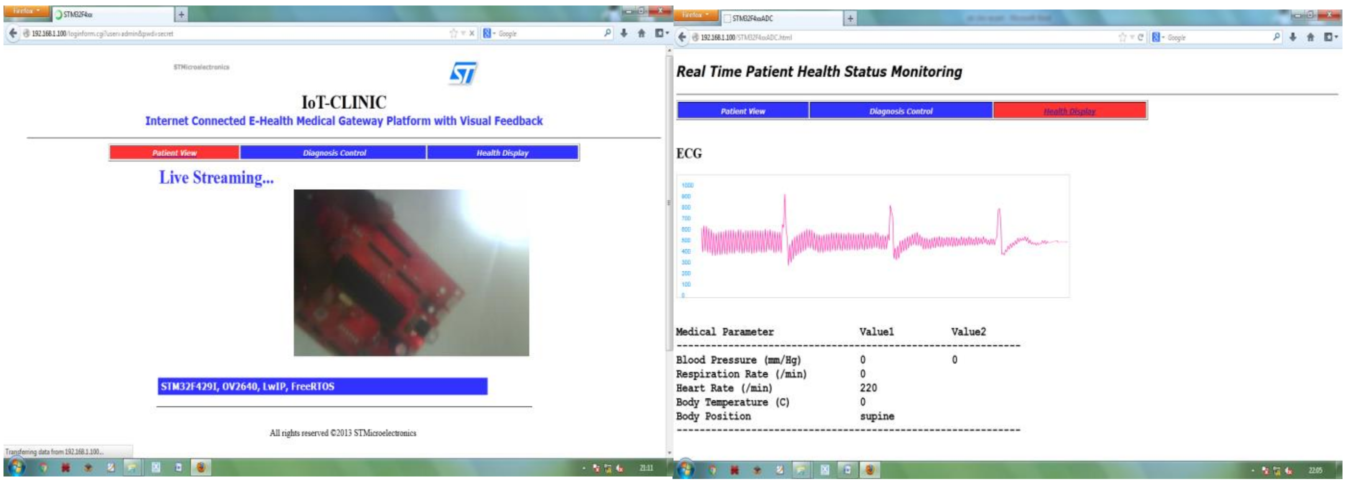


Figure 20 Experimental setup



STMicroelectronics

IoT-CLINIC

Internet Connected E-Health Medical Gateway Platform with Visual Feedback

Patient View | Diagnosis Control | Health Display

Live Streaming...

STM32F429I, OV2640, LwIP, FreeRTOS

All rights reserved ©2013 STMicroelectronics

Real Time Patient Health Status Monitoring

Patient View | Diagnosis Control | Health Display

ECG

```
Medical Parameter              Value1        Value2
-------------------------------------------------------------
Blood Pressure (mm/Hg)         0             0
Respiration Rate (/min)        0
Heart Rate (/min)              220
Body Temperature (C)           0
Body Position                  supine
-------------------------------------------------------------
```

Figure 21 fetched output

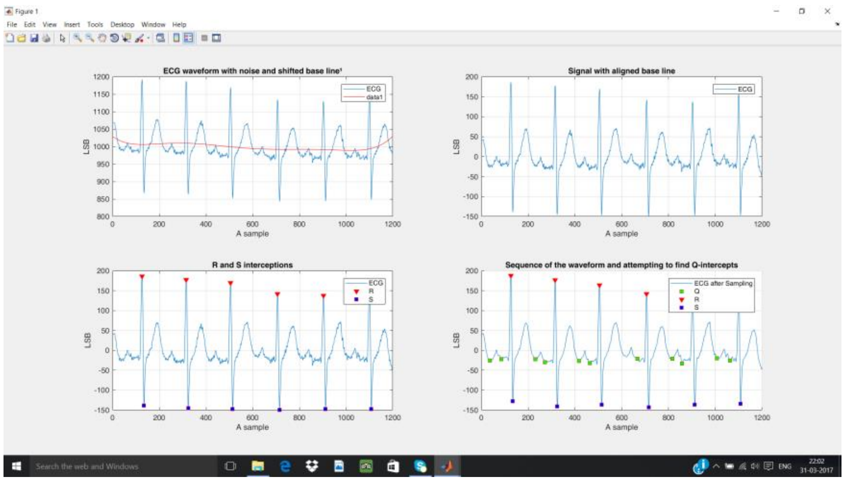


Figure 22 matlab simulation of QRS complex detection

CHAPTER 7
CONCLUSION AND FUTURE WORKS

The results are satisfactorily obtained and the scope for the implementation of artificial intelligence in the system of healthcare monitoring is bright. The website and the application developed can be used for multiple patients monitoring by replacing the embedded architecture with the system on chip architecture. The IoT in build inside server and real time analysis of those parameters are possible developments in the future systems. The system explained above is intimidating and proves to be a starting step in developing a highly intelligent health monitoring system with all real time analytical capabilities. The IoT analytics is the field which requires to be optimized that the real time synthesis of health parameters and/or control parameters could be implemented for better results.

Moreover, the remote areas and rural residents experience barriers to healthcare that limit their ability to get the care they need. Right medical consultation at the right time offered without delay is the difference between prosperity and poverty of a nation. The is a project designed to achieve this mission and is an effort to reduce the health disparities arising out of rural versus urban divide in our country.

CHAPTER 8

ANNEXTURE I

8.1 CODING IMPLEMENTATION FOR STM32F429

```
#include "main.h"
#include "stm32f4xx_hal.h"
/* Middleware */
#include "lwip/netif.h"
#include "lwip/tcpip.h"
#include "cmsis_os.h"
#include "ethernetif.h"
#include "app_ethernet.h"
#include "httpserver-netconn.h"
/* Peripherals */
#include "uart_if_rtos.h"
#include "i2c_if.h"
#include "adc_if.h"
#include "tim_int.h"
/* Components */
#include "led.h"
#include "key.h"
#include "24lc02.h"
#include "bmp180.h"
#include "lis302dl.h"
/* Scheduler includes  */
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "semphr.h"
```

```
/* Standard */
#include <string.h>
/*-------------------------------------------------------------------------*/
/* Constants */
typedef enum _CONTROL_COMMAND_LIST {
        COMMAND_LIST = 1,
        ECG,
        HEART_RATE,
        BLOOD_PRESSURE,
        BODY_POSITION,
        BODY_TEMPERATURE,
        RESPIRATION_RATE
} CONTROL_COMMAND;
/*------------------------------------------------------------------------*/
#define NELEMENTS(array)  (sizeof (array) / sizeof (array[0]))
#define UP_LIMIT         +50
#define DOWN_LIMIT   -50
#define RESPIRATION_THRESH     20
/*------------------------------------------------------------------------*/

static struct netif gnetif;             /* network interface structure */
static struct link_str link_arg;     /* Ethernet link thread Argument */
/*-------------------------------------------------------------------------*/
/* Queue to store control commands from serial console */
xQueueHandle xCtrlCmdQueue = NULL;
/* Mutex to isolate tests done from local and remote threads */
xSemaphoreHandle test_mutex = NULL;
/* Semaphore to signal Ethernet Link state update */
osSemaphoreId Netif_LinkSemaphore = NULL;
osSemaphoreId FrameCaptureSemaphore = NULL;
/*-------------------------------------------------------------------------*/
uint8_t img_buf[61440];  // 60 kB
uint8_t bodyPosStr[16];
uint32_t bodyTemp = 0;
uint32_t respirationRate = 0;
uint32_t heartRate = 0;
int systolic = 0;
int diastolic = 0;
extern volatile uint32_t req_ecgPlotRemote;
extern volatile uint32_t req_heartRateEstimate;
extern volatile uint32_t req_bloodPressureEstimate;
extern volatile uint32_t req_respirationRateEstimate;
extern volatile uint32_t req_bodyPositionEstimate;
extern volatile uint32_t req_bodyTemperatureEstimate;
uint16_t ecg[ECG_SAMPLE_SIZE];
static void SystemClock_Config(void);
static void MX_GPIO_Init(void);
void Error_Handler(void);
static void buzzer_beep_medium(void) {
        buzzer_on();
        vTaskDelay(500);
        buzzer_off();
}
/*-------------------------------------------------------------------------*/
static void buzzer_beep_double(void) {
        buzzer_on();
        vTaskDelay(30);
        buzzer_off();
        vTaskDelay(120);
        buzzer_on();
        vTaskDelay(30);
        buzzer_off();
}
/*---------------------------------------------------------------------------*/

static uint32_t digitalRead_BPM_TriggerButton(void) {
        if (HAL_GPIO_ReadPin(BPM_KEY_GPIO_Port, BPM_KEY_Pin) ==
GPIO_PIN_RESET) {
                vTaskDelay(20);
                if (HAL_GPIO_ReadPin(BPM_KEY_GPIO_Port, BPM_KEY_Pin)
== GPIO_PIN_RESET) {
                        return 1;
                }
        }
        return 0;
}
/*---------------------------------------------------------------------------*/
static int getSystolicPressure(void) {
        return systolic;
}
/*---------------------------------------------------------------------------*/
static int getDiastolicPressure(void) {
        return diastolic;
}
/*---------------------------------------------------------------------------*/

static int32_t getRRPressureSensor(void) {
        int32_t temperature, altitude, pressure;
        bmp180_read(&temperature, &pressure, &altitude);
        return pressure;
}
/*---------------------------------------------------------------------------*/
static void Netif_Config(void) {
        struct ip4_addr ipaddr;
        struct ip4_addr netmask;
        struct ip4_addr gw;
}
/*--------------------------------------------------------------------------*/
static void Peripheral_Init(void) {
        /* Wait for others to pick them up */
        HAL_Delay(100);
        /* Initialize Board LEDs */
        LED_Init();
        /* Initialize KEYs */
        Key_Init();
       /* Initialize TIM3 in Interrupt Mode */
        MX_TIM3_Init();
         /* Initialize ECG Monitor UART */
        MX_USART2_UART_Init(9600);
         /* Initialize Console UART */
        MX_USART3_UART_Init(115200);
         /* Initialize ADC3 (Interrupt mode) */
        MX_ADC3_Init();
         /* Initialize I2C */
        MX_I2C2_Init();
        /* Initialize Accelerometer (Body Position) */
```

```
	lis302dl_init(LOW_G);
	/* Initialize Pressure Sensor (Respiration Rate) */
	bmp180_init();
	/* Initialize OV2640 Camera */
	BSP_CAMERA_Init(RESOLUTION_R480x272, PIXFORMAT_JPEG);
}
/*--------------------------------------------------------------------------*/
static void consoleSendCommandList(void) {
	Serial_print("Command List\r\n");
	Serial_print("ECG\r\n");
	Serial_print("Heart Rate\r\n");
	Serial_print("Blood Pressure\r\n");
	Serial_print("Body Position\r\n");
	Serial_print("Body Temperature\r\n");
	Serial_print("Respiration Rate\r\n");
	Serial_println();
}
/*--------------------------------------------------------------------------*/
static void consoleSendOK(void) {
	Serial_print("ACK: ");
	Serial_print("OK");
	Serial_println();
	Serial_println();
	buzzer_beep_short();
}
/*--------------------------------------------------------------------------*/
static void consoleSendERROR(void) {
	Serial_print("ACK: ");
	Serial_print("Error");
```

```
        Serial_println();
        Serial_println();
        buzzer_beep_medium();
}
/*-----------------------------------------------------------------------*/
static void ControlThread(void const * argument) {
         vTaskDelay(100);
        Serial_print("              IoT-CLINIC");
        Serial_println();
        Serial_println();
        vTaskDelay(10);
         Serial_print("Press BLUE button to start the demo...");
        Serial_println();
        Serial_println();
        vTaskDelay(10);
         // After opening the serial port,
        // press blue button to begin the demo
        while(!BoardKey2) {
                vTaskDelay(50); }
             BoardKey2 = 0;
       if (eeprom_check()) {
                Serial_print("OK");
                Serial_println();
} else {
                Serial_print("EEPROM Check Not OK");
                Serial_println();
               Error_Handler();
        }
        /* Reset diagnostic parameters */
        strcpy((char *) bodyPosStr, "None");
        bodyTemp = 0;
        systolic = 0;
        diastolic = 0;
        heartRate = 0;
        respirationRate = 0;
        uint8_t cmd_error = 0;
        CONTROL_COMMAND ctrl_cmd;
         Serial_println();
         while (1) {
                Serial_print("CMD: ");
                Serial_read_String();       // Terminate with CR+LF
                Serial_print((char *) aUart3RxBuffer);
                Serial_println();
                cmd_error = 0;
                if (strcmp((const char *) aUart3RxBuffer, "Command List") == 0) {
                        ctrl_cmd = COMMAND_LIST;
                        consoleSendOK();
                        consoleSendCommandList();
                        LED1_Off();
                } else if (strcmp((const char *) aUart3RxBuffer, "ECG") == 0) {
                        ctrl_cmd = ECG;
                        consoleSendOK();
                } else if (strcmp((const char *) aUart3RxBuffer, "Heart Rate") == 0)
{
                        ctrl_cmd = HEART_RATE;
                        consoleSendOK();
          } else if (strcmp((const char *) aUart3RxBuffer, "Blood Pressure") == 0)
{
```

```
                        ctrl_cmd = BLOOD_PRESSURE;
                        consoleSendOK();
            } else if (strcmp((const char *) aUart3RxBuffer, "Body Position") == 0)
{
                        ctrl_cmd = BODY_POSITION;
                        consoleSendOK();
    } else if (strcmp((const char *) aUart3RxBuffer, "Body Temperature") == 0)
{
                        ctrl_cmd = BODY_TEMPERATURE;
                        consoleSendOK();
        } else if (strcmp((const char *) aUart3RxBuffer, "Respiration Rate") == 0)
{
                        ctrl_cmd = RESPIRATION_RATE;
                        consoleSendOK();
        } else {
                        consoleSendERROR();
                        cmd_error = 1;
                }

                if (!cmd_error) {
                        xQueueSendToBack(xCtrlCmdQueue, &ctrl_cmd,
portMAX_DELAY);// send the command using Queue
                }
        }
}
/* Task that runs the tests requested from the local pc */
static void LocalThread(void const * argument) {
        vTaskDelay(100);
        CONTROL_COMMAND ctrl_cmd;
        while (1) {
                /* Receive the Control Command */
                xQueueReceive(xCtrlCmdQueue, &ctrl_cmd, portMAX_DELAY);
                if (xSemaphoreTake(test_mutex, portMAX_DELAY) == pdPASS)
{       // acquire the mutex
                if (ctrl_cmd == ECG) {
                                ecgPlotSerial();
                        } else if (ctrl_cmd == HEART_RATE) {
                                heartRateEstimate();
                        } else if (ctrl_cmd == BLOOD_PRESSURE) {
                                bloodPressureEstimate();
                        } else if (ctrl_cmd == BODY_POSITION) {
                                bodyPositionEstimate();
                        } else if (ctrl_cmd == BODY_TEMPERATURE) {
                                bodyTemperatureEstimate();
                        } else if (ctrl_cmd == RESPIRATION_RATE) {
                                respirationRateEstimate();
                        }
                        xSemaphoreGive(test_mutex);   // release the mutex
                }
            }
}
/*---------------------------------------------------------------------------*/
/* Task that receive the diagnosis request from the remote end */
static void RemoteThread(void const * argument) {
        vTaskDelay(1000);
        while (1) {
                vTaskDelay(100);
                if (!req_bloodPressureEstimate && !req_bodyPositionEstimate  &&
            !req_bodyTemperatureEstimate && !req_ecgPlotRemote &&
            !req_heartRateEstimate && !req_respirationRateEstimate)
        {
            continue;
        }
        if (xSemaphoreTake(test_mutex, portMAX_DELAY) == pdPASS) {
// acquire the mutex
            buzzer_beep_double();
            if (req_bloodPressureEstimate) {
                req_bloodPressureEstimate = 0;
                bloodPressureEstimate();
            } else if (req_respirationRateEstimate) {
                req_respirationRateEstimate = 0;
                respirationRateEstimate();
            } else if (req_ecgPlotRemote) {
                req_ecgPlotRemote = 0;
                ecgPlotSerial();
          } else if (req_heartRateEstimate) {
                req_heartRateEstimate = 0;
                heartRateEstimate();
          } else if (req_bodyPositionEstimate) {
                req_bodyPositionEstimate = 0;
                bodyPositionEstimate();
          } else if (req_bodyTemperatureEstimate) {
                req_bodyTemperatureEstimate = 0;
                bodyTemperatureEstimate();
          }
           xSemaphoreGive(test_mutex);     // release the mutex
        }
        }
}
/*---------------------------------------------------------------------------*/
static void vCreateSystemQueues(void) {
            xCtrlCmdQueue = xQueueCreate(1, sizeof(CONTROL_COMMAND));
}
/*---------------------------------------------------------------------------*/
void Error_Handler(void) {
        portDISABLE_INTERRUPTS();
        /* Baseboard LED is On */
        LED4_On();
        /* Nucleo LEDs are Off */
        LED1_Off();
        LED2_Off();
        LED3_Off();
        while (1) {
        }
}
/*---------------------------------------------------------------------------*/
```

8.2 MATLAB CODE FOR QRS COMPLEX DETECTION

```
clear all
clc
close all
% The name of the .dat file (must be in the same directory as the script
filename = '215.dat';
fid=fopen(filename,'r');
% Time corresponds to samples / 360 (sampling) / 2 (because there are two
channels)
samples = 2*1200;
f=fread(fid,samples,'ubit12');
% We only take data from one channel (every other sample)
data=f(1:2:length(f));
wczytywanie()
subplot(2,2,1)
plot(data)
title('ECG waveform with noise and shifted base line¹')
grid on
xlabel('A sample')
ylabel('LSB')
legend('ECG')
hold on
% Level alignment
[p,s,mu] = polyfit((1:length(data))',data,6);
y = polyval(p,(1:length(data))',[],mu);
ECG_data = data - y;
plot(y,'-r')
hold off
subplot(2,2,2)
plot(ECG_data); grid on
%ax = axis; axis([ax(1:2) -1.2 1.2])
title('Signal with aligned base line')
xlabel('A sample'); ylabel('LSB')
legend('ECG')
% Graph3 - R + S
subplot(2,2,3)
plot(ECG_data); grid on
title('R and S interceptions')
xlabel('A sample'); ylabel('LSB')
% RS detection
```

```
[~,locs_Rwave] =
findpeaks(ECG_data,'MinPeakHeight',100,'MinPeakDistance',100);
ECG_inverted = -ECG_data;
[~,locs_Swave] =
findpeaks(ECG_inverted,'MinPeakHeight',100,'MinPeakDistance',100);
hold on
plot(locs_Rwave,ECG_data(locs_Rwave),'rv','MarkerFaceColor','r');
plot(locs_Swave,ECG_data(locs_Swave),'rs','MarkerFaceColor','b');
%axis([0 1850 -1.1 1.1]); grid on;
legend('ECG','R','S');
hold off
% FIGURE 4
smoothECG = sgolayfilt(ECG_data,7,21);
[~,min_locs] = findpeaks(-smoothECG,'MinPeakDistance',40);
% Peaks between -0.2mV and -0.5mV
locs_Qwave = min_locs(smoothECG(min_locs)>-35 & smoothECG(min_locs)<-
20);
subplot(2,2,4)
hold on
plot(smoothECG);
plot(locs_Qwave,smoothECG(locs_Qwave),'rs','MarkerFaceColor','g');
plot(locs_Rwave,smoothECG(locs_Rwave),'rv','MarkerFaceColor','r');
plot(locs_Swave,smoothECG(locs_Swave),'rs','MarkerFaceColor','b');
grid on
title('Sequence of the waveform and attempting to find Q-intercepts')
xlabel('A sample'); ylabel('LSB')
%   ax = axis; axis([0 1850 -1.1 1.1])
legend('ECG after Sampling','Q','R','S');
fprintf('The positions of the bracketsQ:\n')
for N=1:length(locs_Qwave)
    fprintf('%d ', locs_Qwave(N))
end
fprintf('\n');
fprintf('The positions of the brackets R:\n')
for N=1:length(locs_Rwave)
    fprintf('%d ', locs_Rwave(N))
end
fprintf('\n');
fprintf('The positions of the brackets S:\n')
for N=1:length(locs_Swave)
    fprintf('%d ', locs_Swave(N))
end
end
```

REFERENCES

[1]. P.A. Laplante and N. Laplante, “A Structured Approach for Describing Healthcare Applications for The Internet of Things,” *Proc. IEEE 2nd World Forum on Internet of Things*, to appear, 2016.

[2]. L. Catarinucci et al., “An IoT-Aware Architecture for Smart Healthcare Systems,” *IEEE Internet of Things J.*, vol. 2, no. 6, 2015, pp. 515–526.

[3]. K. Sato et al., “Feasibility Assessment of Bluetooth-Based Location System for Workflow Analysis of Nursing Staff,” *Trans. Japanese Society for Medical and Biological Eng.*, vol. 51, supplement, 2013, p. R-314.

[4]. “Nurses Rank as Most Honest, Ethical Profession for 14th Straight Year,” press release, Am. Nurses Assoc. (ANA), 2015.

[5]. D. Skiba, “Emerging Technologies: The Internet of Things (IoT),” *Nursing Education Perspectives*, vol. 34, no. 1, 2013, pp. 63–64.

[6]. X. Liang *et al.*, “Enabling Pervasive Healthcare through Continuous Remote Health Monitoring,” *IEEE Wireless Commun.*, vol. 19, no. 6, Dec. 2012, pp. 10–18.

[7]. A. Toninelli, R. Montanari, and A. Corradi, “Enabling Secure Service Discovery in Mobile Healthcare Enterprise Networks,” *IEEE Wireless Commun.*, vol. 16, no. 3, June 2009, pp. 24 32 .

[8]. J.Zhou *et al.*, “Securing m-Healthcare Social Networks: Challenges, Countermeasures, and Future Directions,” *IEEE Wireless Commun.*, vol. 20, no. 4, Aug. 2013, pp. 12–21.

[9]. H. Wang *et al.*, “Resource-Aware Secure ECG Healthcare Monitoring Through Body Sensor Networks,” *IEEE Wireless Commun.*, vol. 17, no. 1, Feb. 2010, pp. 12–19.