REAL TIME STRESS ANALYSIS AND DETECTION SYSTEM USING GSR SENSOR

A PROJECT REPORT

Submitted by

G.HARSHAVARDHINI -412519106044
N.PRIYADARSHINI -412519106109

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI SAIRAM ENGINEERING COLLEGE
(An Autonomous Institution; Affiliated to Anna University, Chennai -600 025)
ANNA UNIVERSITY :: CHENNAI 600 025

`

APRIL 2023

SRI SAIRAM ENGINEERING COLLEGE

(An Autonomous Institution; Affiliated to Anna University, Chennai -600 025)

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"REAL TIME STRESS ANALYSIS AND DETECTION SYSTEM USING GSR SENSOR"** is the bonafide work of **" HARSHVARDHINI G (412519106044) PRIYADARSHINI N (412519106109)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr J.RAJA	**Mr S.VINOTH KUMAR**
PROFESSOR	**ASSISTANT PROFESSOR**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Electronics and	Department of Electronics and
Communication Engineering	Communication Engineering
Sri Sai Ram Engineering College,	**Sri Sai Ram Engineering College,**
(Autonomous) Chennai-600 044.	**(Autonomous) Chennai-600 044.**

Submitted for the project viva-voce examination held on ___________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Nowadays, it is almost impossible to imagine that anyone can live without computers. Computers are important due to the significant improvements they have made in work efficiency and productivity. Working with computers can be stressful, especially when a person is exposed to it for a long period of time. Knowing one’s stress level while using the computer can help the person adjust his time on the computer. This research aims to help everyone become more aware of their stress levels during their computer usage by the use of an algorithm that can detect stress levels. A series of interviews were conducted to further determine the appropriate data required in the formulation of the algorithm. Experts from various fields were interview respondents for the pre-development phase. On the other hand, the faculty and students of the Computer Science Department of the University of the San Carlos were the respondents for the testing in the post-development phase. The system was tested using data gathered from Galvanic Skin Response (GSR) sensors. The gathered data were used to detect the stress and updated in iot.

ACKNOWLEDGEMENT

We thank our Founder Chairman **Thiru. MJF . Ln. LEO MUTHU** for his great endeavors in establishing this institution and standing as a figure of guidance.

We also thank our **CEO Dr J. SAI PRAKASH LEO MUTHU** and Principal **Dr K. PORKUMARAN** for their kind cooperation and inspiration.

We thank **Dr J. RAJA**, Head of the Department, Electronics and Communication Engineering for giving us the freedom to carry out the project work in the chosen domain.

We thank our Project Coordinator **Dr. B. PANJAVARNAM,** Associate professor , and **Mr. S . VINOTH KUMAR,** department of ECE Assistant professor for their constant support right from the commencement of the project work and also for providing us necessary details with regard to presentation and documentation.

We are ever grateful to our Project Guide **Mr. S.VINOTH KUMAR,** Assistant professor who was a buttress to carry out our project and for his valuable suggestions at every stage of the project. We sincerely thank him for the support rendered from the day wecommenced our work.

Our gratitude extends to the **Staff of the ECE Department** whose words of encouragement kept our spirits high throughout the course of the project.

We thank the entire people who contributed directly and indirectly for the completion of the project.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Stress is a body's method for reacting to a challenge. Human stress can have an impact on a person's mental and physical well - being. Stress can lead to a change in behavior and in physiology. Many people suffer from stress in everyday life. Stress is related to human work in one way or other. Originates of stress have different sources such as time pressure while working in company, responsibility, economic problem or physical factors such as noise. Signs of stress are human fell tension, anxious, angry, frustrated or irritated by things over which he has no control. Stress detection is an on-going research topic among both psychologists and engineers. Wearable sensors and bio signal processing technologies are developed for detecting the human stress. The physiological signals proposed by this project are Galvanic Skin Response (GSR), also known as Skin Conductance (SC). These signals were selected based on their properties regarding non-invasively when being acquired and because of their variation is strongly related to stress stimuli.

1.1 EMBEDDED SYSTEM

An embedded system is a microprocessor-based computer hardware system with software that is designed to perform a dedicated function, either as an independent system or as a part of a large system. At the core is an integrated circuit designed to carry out computation for real-time operations.Complexities range from a single microcontroller to a suite of processors with connected peripherals and networks; from no user interface to complex graphical user interfaces. The complexity of an embedded system varies significantly depending on the task for which it is designed.Embedded system applications range from digital watches and microwaves to hybrid vehicles and avionics. As much as 98 percent of all microprocessors manufactured are used in embedded systems.

Embedded systems are managed by microcontrollers or digital signal processors (DSP), application-specific integrated circuits (ASIC), field-programmable gate arrays (FPGA), GPU technology, and gate arrays. These processing systems are integrated with components dedicated to handling electric and/or mechanical interfacing.Embedded systems programming instructions, referred to as firmware, are stored in read-only memory or flash memory chips, running with limited computer hardware resources. Embedded systems connect with the outside world through peripherals, linking input and output devices.

1.2 DEVELOPMENT OF EMBEDDED SYSTEM

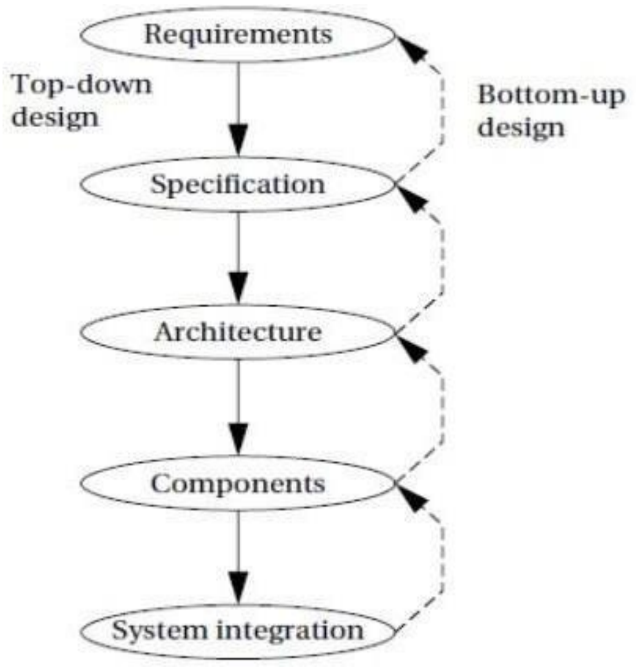


Fig 1.1 Development of embedded systems

An embedded system design process is how a manufacturer determines the requirements for a small computerized system embedded within a product. Then, they decide the best way to build that system and test that it works. In an embedded system, hardware and software work together.

Embedded systems consist of a microcontroller with on-board memory, a power supply, and communication ports for transmitting data to other devices. Embedded software programs tell the microcontroller how to respond in real time to data collected from the environment through peripheral sensors and devices.

Embedded systems are designed to perform specific tasks against specific resource constraints which could include memory, code space, processor cycles or peripherals, or battery life. These are interchangeable insofar as every project will determine its own priorities.

For some projects, most commonly in applications such as medical, military and aerospace, with high thresholds of quality and reliability requirements, bugs or system failure can be life-endangering; a malfunctioning medical device used during surgery could be fatal.

Changing the embedded system configuration (hardware and software) after installation is challenging and often costly. The initial selection of software platform and other system features becomes critical across the embedded system design and development process – creating the architecture, implementing the architecture, testing the system, maintaining the system and enabling future development activity such as system scaling in all directions.

The challenge for developers is to enable seamless connectivity of the device, ensuring performance and reliability against security risks within operating environments. Determining the security requirements has become critical in the context of the overall system and eventual operating environments. This will help to achieve a balance between functionality and security.

The most important step in the development of embedded systems is the right choice of a software platform. This enables future development and/or scaling of the system without changing the core technology.

1.3 Internet Of Things

The Internet of things (IoT) describes physical objects (or groups of such objects) with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks. Internet of things has been considered a misnomer because devices do not need to be connected to the public internet, they only need to be connected to a network and be individually addressable.

The field has evolved due to the convergence of multiple technologies, including ubiquitous computing, commodity sensors, increasingly powerful embedded systems, and machine learning.Traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), independently and collectively enable the Internet of things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the "smart home", including devices and appliances (such as lighting fixtures, thermostats, home security systems, cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and smart speakers. IoT is also used in healthcare systems.

There are a number of concerns about the risks in the growth of IoT technologies and products, especially in the areas of privacy and security, and consequently, industry and governmental moves to address these concerns have begun, including the development of international and local standards, guidelines, and regulatory frameworks.

1.4 APPLICATIONS OF EMBEDDED SYSTEM AND IoT

Applications of Embedded System:

- Body or Engine safety.
- Entertainment and multimedia in car.
- E-Com and Mobile access.
- Robotics in assembly line.
- Wireless communication.
- Mobile computing and networking
- Digital cameras.

- Digital wristwatches.
- Appliances, such as refrigerators, washing machines, and microwave ovens.
- Temperature measurement systems.
- Thyristor Power Control with IR Remote
- Auto Power Supply Control from 4 Different Sources: Solar, Mains, Generator &Inverter to ensure no break power
- Lamp Life Extender by ZVS (Zero Voltage Switching)
- Automatic Plant Irrigation System on Sensing Soil Moisture Content
- Speed Synchronization of Multiple Motors in Industries

Applications of IoT:

IoT is essentially a platform where embedded devices are connected to the internet, so they can collect and exchange data with each other. It enables devices to interact, collaborate and, learn from each other's experiences just like humans do. The Internet carries many applications and services, most prominently the World Wide Web, including social media, electronic mail, mobile applications, multiplayer online games, Internet telephony, file sharing, and streaming media services.

CHAPTER 2 LITERATURE SURVEY

There are many methods of real time stress detection system using gsr sensor

2.1 Detection of Stress in Human Brain, Prithwijit Mukherjee, Dr.Anisha Halder Roy Prithwijit Mukherjee, Dr.Anisha Halder Roy,2011 IEEE International Conference On Bioinformation and Biomedicine

The main goal of the study is to create a stress detection mechanism and a stress level indicator circuit for electroencephalogram (EEG) signal-based stress level measurement of the human brain. For measuring stress, signals from the frontal lobe of the human brain have been employed. Thirty participants' brainwaves are monitored as they go through five sets of increasingly difficult mathematical problems. In completing these question sets, we suppose that the subjects experience five various stress levels, including "Relaxed," "Less stressed," "Moderately Stressed," "Hyper Stressed," and "Alarmingly Stressed." Following that, characteristics are retrieved from the processed recorded EEG data. We create a feed- forward neural network to categorise the level of stress in the human brain.

2.2 Stress Detection Via Sensor Translation, Sirat Samyoun, Abu Sayeed Mondol,2020 16th International Conference on Disturbuted Computing in Sensor System

Anxiety, hypertension, and cardiovascular illnesses are just a few of the mental and physical health issues that stress makes more likely. If stress can be continuously evaluated, better advice and strategies for reducing its effects can be given. The capacity to assess a variety of physiological signals related to stress plus the recent rise in wearable technology have made it possible to continuously measure stress in the wild. Most wearable devices for physiological signal measurement are worn on the wrist and chest. Although wrist sensors alone do not now yield the same level of accuracy in detecting stress as chest sensors do, chest devices are less common and convenient overall, especially in the free-living setting. In this article, we offer a technique for utilising.

2.3 User Independent Human Stress Detection, Ramanathan Murugappan, Joish J Bosco,2020 IEEE 10th International Conference on Intelligent System

Stress has a significant impact on a person's capacity for making decisions, paying attention, learning, and problem-solving. In recent years, research in the domains of psychology and computer science has focused heavily on stress detection and modelling. Affective states, which are the sensations of feeling the underlying emotional state, are used by psychologists to measure stress. The majority of the classification of human stress was done using user-dependent models that couldn't be generalised to a different user. As a result, training the model to anticipate the user's affective states takes a lot of time for a new user. The authors of this work provide a user-independent classification model for detecting human stress in which a new user doesn't need to have their affective state calibrated beforehand.

2.4 A Stress Detection System Based on Multimedia Input Periperals, L. Ciabattoni, G. Foresi, F. Lamberti,2020 IEEE International Conference on Consumer Electronic

This study presents a Stress Detection System based on keyboard and mouse data and Machine Learning Algorithms (MLAs). There are three steps in the creation of this system. First, a web application framework gathers data from the keyboard and mouse while each user does some actions. He or she conveys the stress level at the conclusion of each assignment in order to establish the stress class. Second, a Neighborhood Component Analysis (NCA) is used to conduct features extraction and features selection methods from the gathered data. Finally, three MLAs are used to detect stress. These MLAs were trained with features as input and stress classes as output.

2.5 Smart Wearable and for Stress Detection, Muhamma Zubair , 2015 5th International Conference on IT Convergence and Security

Mental tension must occasionally be managed because it can cause a variety of dangerous suffering. Early identification of mental stress can aid in preventing stress-related health issues. The purpose of this study is to develop a wearable, low-cost, IoT-based smart band for health care that can identify mental stress based on changes in skin conductance. This device may wirelessly communicate data linked to stress to the user's smartphone while also continuously monitoring the user's mental stress. It not only aids the customers in better comprehending their stress patterns but also offers the doctor trustworthy information for a superior course of treatment. This device receives a variety of signals from various sensors as inputs. This band makes predictions by carefully examining the association between these signals using a machine learning technique.

CHAPTER 3
PROBLEM FORMULATION AND SOLUTION

3.1 PROBLEM STATEMENT:

Stress is a widespread issue that has an impact on people's health, happiness, and productivity. Many physical and mental health issues, including as high blood pressure, heart disease, anxiety, and melancholy, may result from it. There is still a sizable gap in the capacity to identify and monitor stress levels in real-time, despite the growing knowledge of the harmful effects of stress. Individuals find it challenging to successfully regulate their stress levels as a result, which lowers quality of life and raises healthcare costs.

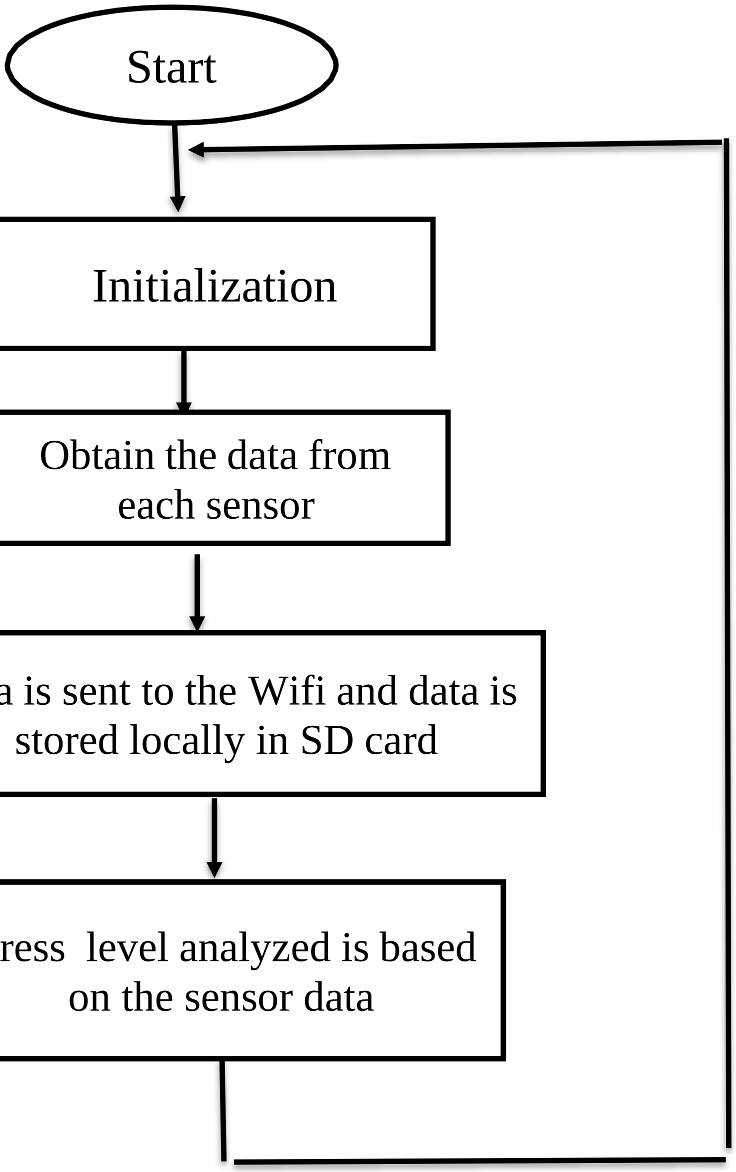
- The goal of real-time stress analysis is to create a system that can identify and track stress levels precisely in real-time while also giving users quick feedback and tips on how to successfully manage their stress.
- The device should be non-intrusive, simple to operate, and portable so that people can wear it for long periods of time without feeling uncomfortable.

3.2 PROPOSED SOLUTION:

Real-time stress analysis using GSR sensor involves developing a system that can accurately detect and monitor stress levels in real-time, providing individuals with immediate feedback and intervention strategies to manage their stress levels. The solution has significant implications for improving people's health and well-being, leading to a happier, healthier, and more productive life.

3.3 FLOW DIAGRAM

Fig.3.1 Block diagram



Start
Initialization
Obtain the data from each sensor
Data is sent to the Wifi and data is stored locally in SD card
Stress level analyzed is based on the sensor data

3.4 PROJECT FEATURES:

- Low Cost
- Efficient and Effective
- Developed into a product successfully
- Real-time monitorin

CHAPTER 4

METHODOLOGY

4.1 PROPOSED SYSTEM

We are proposing a system to detect stress continuously . The proposed system contains a microcontroller with GSR sensor. It will monitor the stress level continuously. If the stress level is abnormal, then microcontroller plays the relax music and updated in IOT webpage**.**

4.2 SENSING STAGE

In order to monitor blood pressure, a wearable device, such as a smartwatch or fitness tracker, is used to measure the person's heart rate. An acute stress reaction to a particular incident or circumstance might be indicated by an increase in blood pressure**.**

4.3 DATA ACQUISITION STAGE

Real-time stress analysis includes gathering information on people's stress levels in real-time. a few techniques for real-time stress analysis data gathering. By studying people's behaviour for indications of stress observation can be used to gather information about people's levels of stress**.**

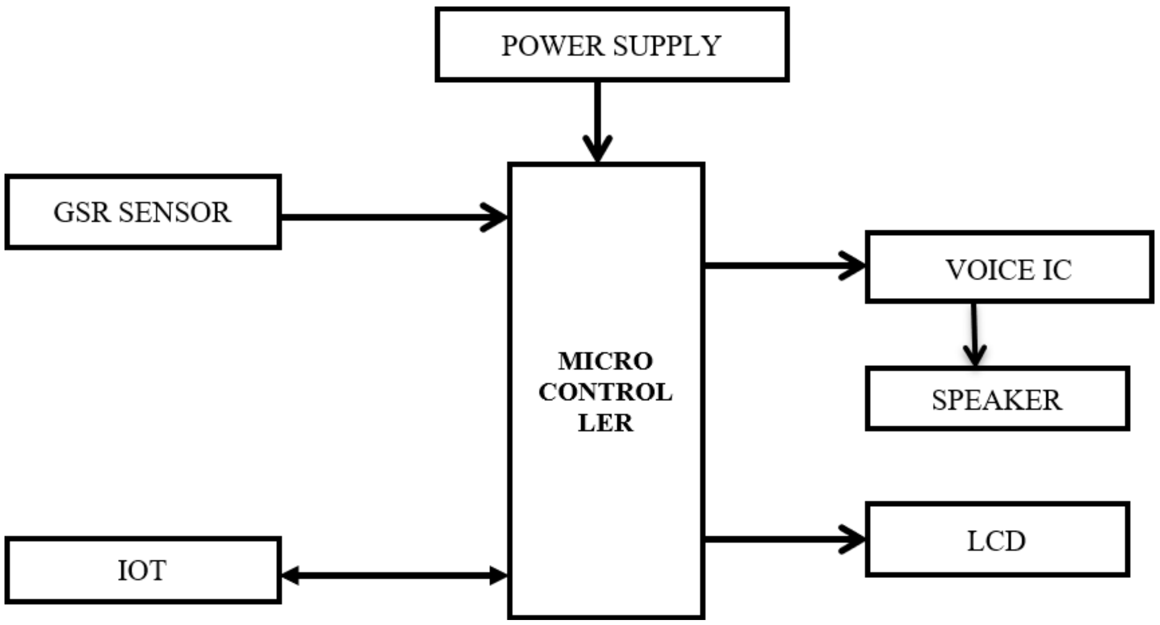


Fig 4.1 Hardware Block diagram

4.5 SYSTEM HARDWARE DESGIN

HARDWARE COMPONENTS:

- Dc Adaptor
- Micro Controller
- Voice IC
- Lcd Display
- GSR Sensor

4.5.1 DC ADAPTOR

Fig 4.2 Dc Adaptor

An electric power adapter may enable connection of a power plug, An AC adapter, also called a "recharger", is a small power supply that changes household electric current from distribution voltage) to low voltage DC suitable for consumer electronics. Some modify power or signal attributes, while others merely adapt the physical form of one electrical connector to another.

FEATURES

- Output current:1A
- Supply voltage: 220-230VAC
- Output voltage: 12VDC
- ·Reduced costs
- Increased value across front-office and back-office functions
- Access to current, accurate, and consistent data
- It generates adapter metadata as WSDL files with J2CA extension.

APPLICATIONS

- Back-end systems which need to send purchase order data to oracle applications send it to the integration service via integration server client.
- SMPS applications.

4.5.2 MICROCONTROLLER ESP32

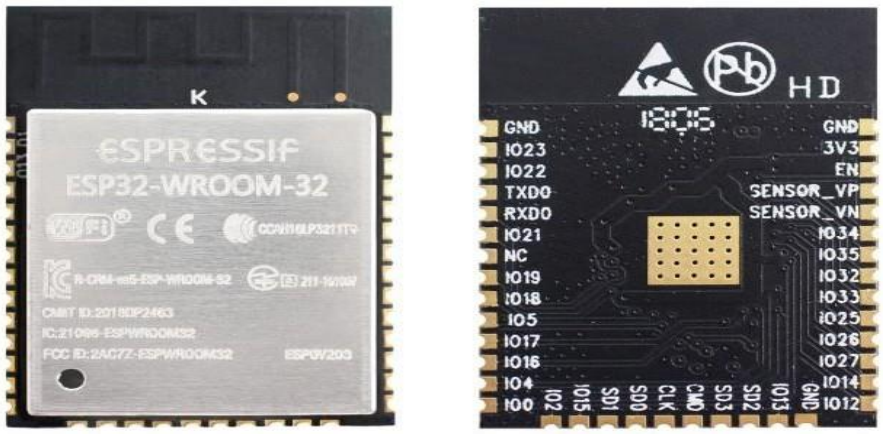


Fig. 4.3 Esp32 Module

ESP32 is a low-cost System on Chip (SoC) Microcontroller from Espressif Systems, the developers of the famous ESP8266 SoC. It is a successor to ESP8266 SoC and comes in both single-core and dual-core variations of the Tensilica’s 32-bit Xtensa LX6 Microprocessor with integrated Wi-Fi and Bluetooth. The good thing about ESP32, like ESP8266 is its integrated RF components like Power Amplifier, Low-Noise Receive Amplifier, Antenna Switch, Filters and RF Balun. This makes designing hardware around ESP32 very easy as you require very few external components.

SPECIFICATIONS OF ESP32 BOARD:

520 KB of SRAM, 448 KB of ROM and 16 KB of RTC SRAM.

- Supports 802.11 b/g/n Wi-Fi connectivity with speeds up to 150 Mbps.
- Support for both Classic Bluetooth v4.2 and BLE specifications.
- 34 Programmable GPIOs.
- Up to 18 channels of 12-bit SAR ADC and 2 channels of 8-bit DAC
- Serial Connectivity include 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART.
- Ethernet MAC for physical LAN Communication (requires external PHY).
- 1 Host controller for SD/SDIO/MMC and 1 Slave controller for SDIO/SPI.
- Motor PWM and up to 16-channels of LED PWM.
- Secure Boot and Flash Encryption.
- Cryptographic Hardware Acceleration for AES, Hash (SHA-2), RSA, ECC and RNG.

4.5.3.VOICE IC-APR9600

The APR9600 device offers true single-chip voice recording, non-volatile storage, and playback capability for 40 to 60 seconds.

The device supports both random and sequential access of multiple messages. Sample rates are user-selectable, allowing designers to customize their design for unique quality and storage time needs. Integrated output amplifier, microphone amplifier,

and AGC circuits greatly simplify system design. The device is ideal for use in portable voice recorders, toys, and many other consumerand industrial applications.

APLUS integrated achieves these high levels of storage capability by using its proprietary analog/multilevel storage technology implementing an advanced Flash non- volatile memory process, where each memory cell can store 256 voltage levels. This technology enables the APR9600 device to reproduce voice signals in their natural form. It eliminates the need for encoding and compression, which often introduce distortion

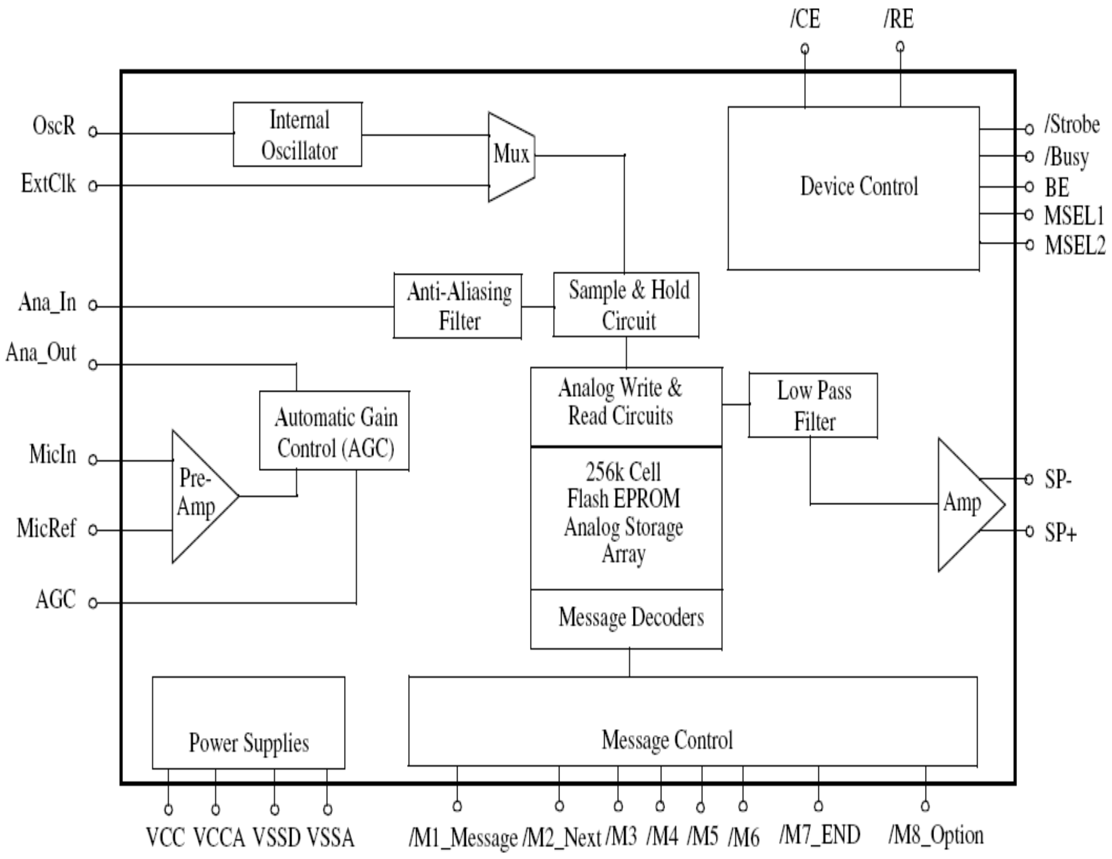


Fig 4.4 Voice IC

PIN DIAGRAM OF APR9600

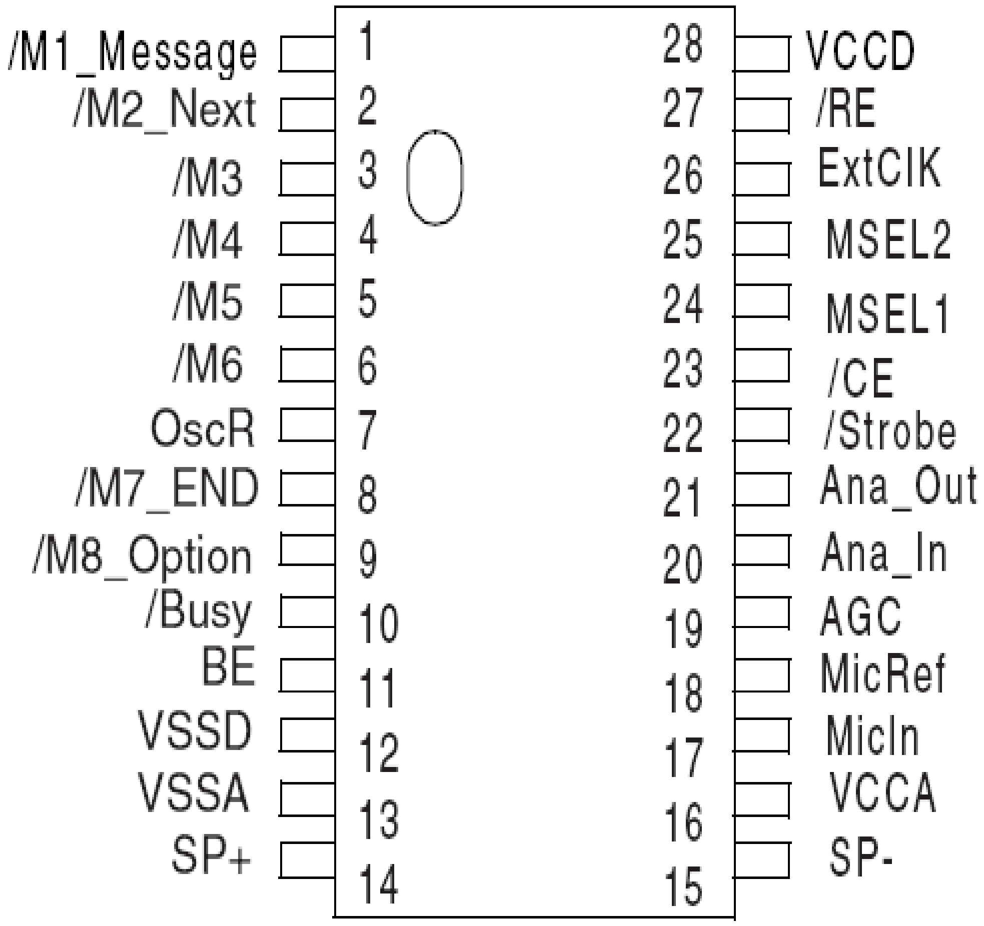


Fig.4.5 Pin diagram

The APR9600 i internal architecture includes an analog inputs, a differential microphone amplifier, and an internal anti-aliasing filter. The signal is then clocked into the memory array through the Sample and Hold circuit and the Analog Write/Read circuit. When playback is desired, the previously stored recording is retrieved from memory, low pass filtered, and amplified. Chip-wide management is accomplished through the device control block and message management is controlled through the message control block. More detail on device application can be found in the Sample Applications section.

Message Management:

Playback and record operations are managed by on chip circuitry.There are several available messaging modes depending upon desired operation. These message modes determine message management style, message length, and external parts count. Therefore, the designer must select the appropriate operating mode before beginning the design. Operating modes do not affect voice quality; for information on factors affecting quality refer to the Sampling Rate & Voice Quality section.

The device supports three message management modes (defined by the MSEL1, MSEL2 and /M8_Option pins shown in Figures 1 and 2):

• Random access mode with 2, 4, or 8 fixed-duration messages

• Tape mode, with multiple variable-duration messages.

Modes cannot be mixed. Switching of modes after the device has recorded an initial message is not recommended. If modes are switched after an initial recording has been made some unpredictable message fragments from the previous mode may remain present, and be audible on playback, in the new mode. These fragments will disappear after a record operation in the newly selected mode. Table 1 defines the decoding necessary to choose the desired mode.

An important feature of the APR9600 message management capabilities is the ability to audibly prompt the user to changes in the device's status through the use of "beeps" superimposed on the device's output. This feature is enabled by asserting a logic high level on the BE pin

Random Access Mode:

Random access mode supports 2, 4, or 8 messages segments of fixed duration. As suggested recording or playback can be made randomly in any of the selected messages. The length of each message segment is the total recording length available (as defined by the selected sampling rate) divided by the total number of segments enabled (as decoded in Table1). Random access mode provides easy indexing to message segments.

Functional Description of Recording in Random Access Mode:

The device is ready to record or playback in any of the enabled message segments as soon as it is powered on. Both /RE and /CE must be set low in order to enable recording and enable the device, respectively. Applying a low level to the message trigger pin that corresponds to the message segment you want to use will start recording. For message segments 1 through 8, the message trigger pins are labelled /M1 Message - /M8 Option on pins 1 through 9 (except pin 7).

The gadget reacts with a single beep at the speaker outputs to signal that it has started recording (if the BE pin is high to enable the beep tone).

As long as the message pin is low, recording will keep going.

Functional Description of Playback in Random Access Mode:

The device is ready to record or replay in any of the enabled message segments as soon as it is powered on. The device must be enabled with /CE set low and recording must be turned off with /RE set high in order to enable playback. Applying a high to low edge to the message trigger pin that corresponds to the message segment you want to playback starts the playback process. Playback won't stop till the message's conclusion. Playback of the current message is terminated instantly if a high to low edge happens on the same message trigger pin while it is being played back.Playback of the current message ends instantly (marked by one beep) and the new message segment starts if a different message trigger pin pulses during playback.

Tape Mode:

Similar to how conventional cassette tape recorders manage messages sequentially, tape mode does the same. There are two choices in tape mode: auto rewind and normal. When in auto rewind mode, the device is set up to automatically go back to the message's beginning after recording or playing it again. Similar to a conventional cassette tape recorder, messages in tape mode must be recorded or played back sequentially.

Function Description Recording in Tape Mode using the Normal option:

On power up, the device is ready to record or play back, starting at the first address in the memory array. To record, /CE must be set low to enable the device and /RE must be set low to enable recording. A falling edge of the /M1_Message pin initiates voice recording (indicated by one beep).

A subsequent rising edge of the /M1_Message pin during recording stops the recording (also indicated by one beep). If the /M1_Message pin is held low beyond the end of the available memory, recording will stop automatically (indicated by two beeps). The device will then assert a logic low on the /M7_END pin for a duration equal to 1600 cycles of the sample clock, regardless of the state of the /M1_Message pin.

The device returns to standby mode when the /M1_Message pin goes high again. After recording is finished the device will automatically rewind to the beginning of the most recently recorded message and wait for the next user input.

Function Description of Playback in Tape Mode using the Normal Option:

The /CE input must be set to low to enable the device and /RE must be set to high to disable recording. The first pulse of the /M1_Message pin initiates playback from the beginning of the current message, with a 1,530 ms period of silence inserted during looping. In auto rewind mode, the device always rewinds to the beginning of the current message. To fast forwarded past the current message, the /M2_Next pin must be low for 400 cycles of the sampling clock. The user can initiate playback of a message with the playback sequence described above.

Functional Description of Recording in Tape Mode using Auto Rewind Option:

The device is ready to record or play back, starting at the first address in the memory array. To begin recording, the /CE input must be set to low and /RE must be set to low. On a falling edge of the /M1_Message pin, the device will beep once and initiate recording. If the /M1_Message pin is held low beyond the end of the available memory, recording stops automatically and two beeps are inserted. To record over all previous messages, the device must pulse the /CE pin low once to reset the device to the beginning of the first message.

Signal Storage:

The APR 9 6 0 0 samples incoming voice signals and stores the instantaneous voltage samples in non-volatile FLASH memory cells. Each memory cell can support voltage ranges from 0 to 256 levels. These 256 discrete voltage levels are the equivalent of 8-bit (28=256) binary encoded values. During playback the stored signals are retrieved from memory, smoothed to form a continuous signal, and then amplified before being fed to an external speaker

Sampling Rate & Voice Quality:

- According to the Shannon's sampling theorem, the highest possible frequency component introduced to the input of a sampling system must be equal to or less than half the sampling.
- The APR9600 automatically filters its input, based on the selected sampling frequency, to meet this requirement.
- Higher sampling rates increase the bandwidth and hence the voice quality, but they also use more memory cells for the same length of recording time.
- The A P R 9 6 0 0 accommodates sampling rates as high as 8 kHz and as low a 4 kH.

Sampling Application:

Reference schematics show how a recording system can be designed. the device configured in tape mode, normal operation. Sampling rate is determined by the resistor value on pin 7, and the RC network on pin 19 sets the AGC "attack time". A bias must be applied to the electret microphone to power its circuitry, and both pins 18 and 19 must be AC couple to the microphone network to block the DC biasing voltage.

Features:

- Single-chip, high-quality voice recording & playback solution
- No external ICs required
- Minimum external components
- Non-volatile Flash memory technology
- No battery backup required
- User-Selectable messaging options

• Random access of multiple fixed-duration messages

• Sequential access of multiple variable-duration messages

• User-friendly, easy-to-use operation

• Programming & development systems not required

• Level-activated recording & edge-activated play back switches

• Low power consumption

• Operating current: 25 mA typical

• Standby current: 1 uA typical

• Automatic power-down

• Chip Enable pin for simple message expansion

4.5.4 LCD DISPLAY

Fig 4.6 Lcd display

LCD Display designed for E-blocks. It is a 16 character, 2-line alphanumeric LCD display connected to a single 9-way D-type connector. It allows the device to be connected to most E-Block I/O ports. It requires data in a serial format, which is detailed in the user guide below. The display also requires a 5V power supply.

FEATURES

• Input voltage: 5v

• E-blocks compatible

• Low cost

• Compatible with most I/O ports in the E-Block range

• Ease to develop programming code using Flow code icons

APPLICATIONS

• Monitoring.

4.5.5 GSR SENSOR

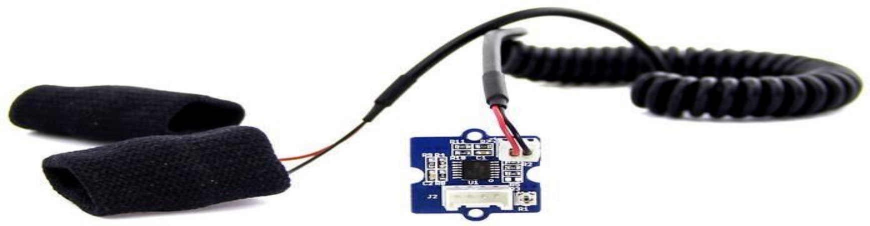


Fig 4.7 GSR Sensor

- GSR sensors measure stress levels and emotional spikes by measuring sweat gland activation.

- Fine-tuning the sensor can be challenging, so results should be evaluated with this in mind.
- Seed Studio's GSR sensor module is adjustable with an onboard potentiometer.
- The sensor will output an analog signal that corresponds to the person’s skin conductance.
- The conductive voltage of skin indicates different states of person's emotions viz. stressed or relaxed. The conductive voltage depends on sweat produced by the sweat glands which are controlled by nervous system. The conductivity of the skin changes due to sweat secretion and consecutively its voltage. The voltage is measured to determine states of emotions of human beings.

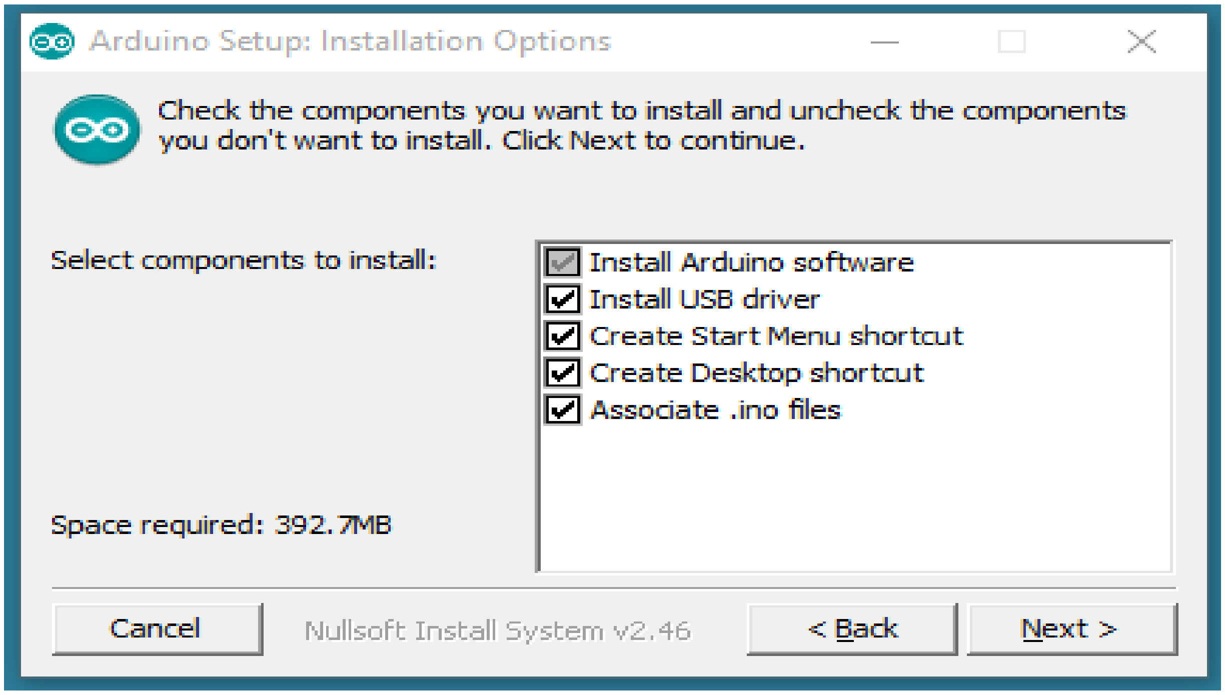
SYSTEM SOFTWARE DESGIN

4.6. SOFTWARE COMPONENTS:

- ARDUINO SOFTWARE (IDE)
- EMBEDDED C

4.6.1 ARDUINO SOFTWARE (IDE)

Get the latest version from the download page. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.


Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

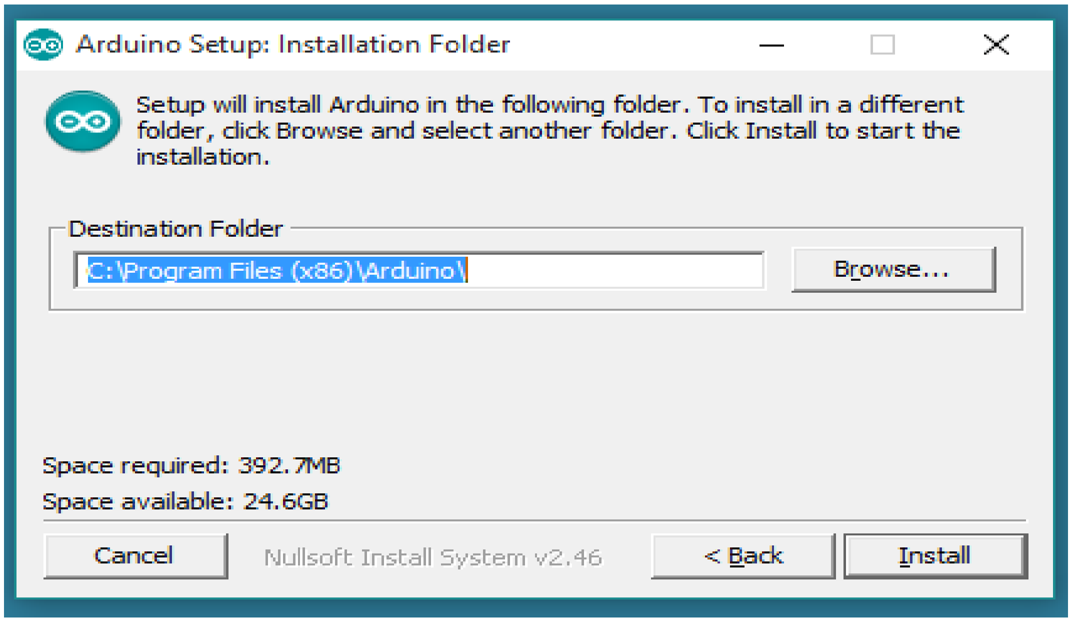
Select components to install:

- [x] Install Arduino software
- [x] Install USB driver
- [x] Create Start Menu shortcut
- [x] Create Desktop shortcut
- [x] Associate .ino files

Space required: 392.7MB

Fig 4.8

Choose the components to install


Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino\

Space required: 392.7MB
Space available: 24.6GB

Fig 4.9

Choose the installation directory (we suggest to keep the default one)

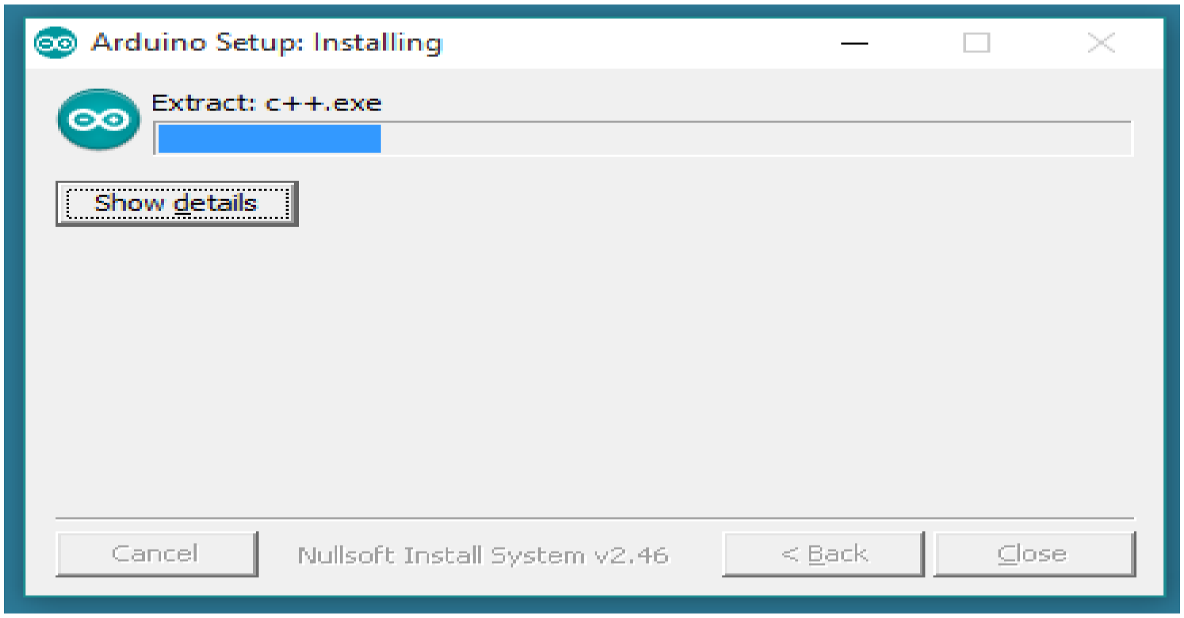


Fig 4.10

The process will extract and install all the required files to execute properly the Arduino Software (IDE).

Arduino Boot loader Issue:

The current boot loader burned onto the Arduino UNO is not compatible with ROBOTC. In its current form, you will be able to download the ROBOTC Firmware to the ArduinoUNO, but you will not able to download any user programs**.**

Arduino UNO firmware prevents flash write commands from starting at the beginning of flash memory.

Re-burn bootloader using Arduino Open Source language with modified bootloader file, backwards compatible with original bootloader.

Hardware Needed:

To burn a new version of the Arduino boot loader to your UNO, you'll need an AVR ISP Compatible downloader.

Using an AVR ISP (In System Programmer):

Your Arduino UNO (to program)

- An AVR Programmer such as the AVR Pocket Programmer
- An AVR Programming Cable (the pocket programmer comes with one)

Using another Arduino as an ISP:

- Your Arduino UNO (to program)
- A Working Arduino (doesn't matter what kind) Some Male-to-Male Jumper Cables

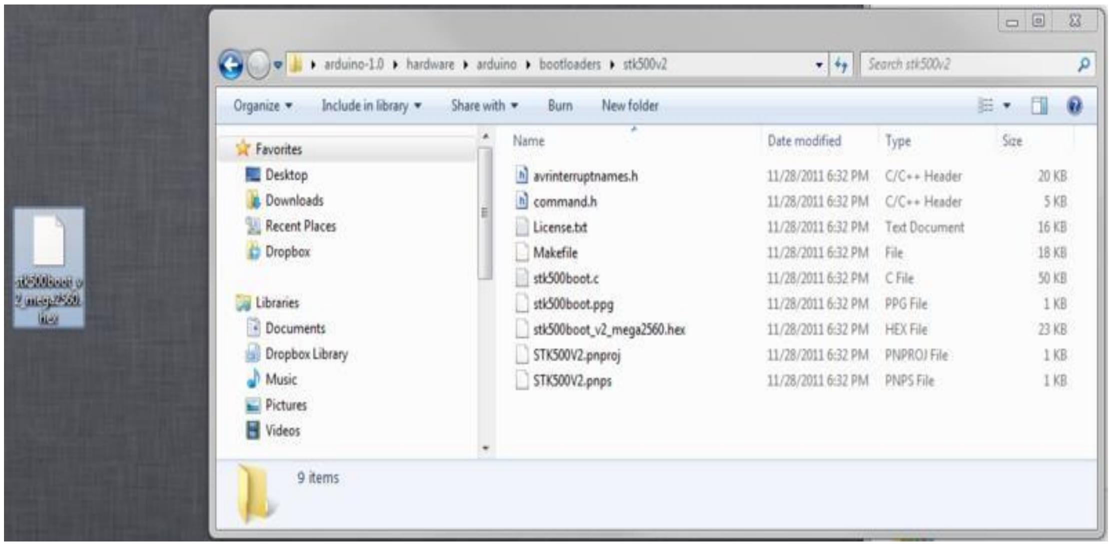
Software Needed:

ROBOTC is not currently able to burn a bootloader onto an Arduino board, so you'll need to download a copy of the latest version of the Arduino Open-Source programming language.

- Arduino Official Programming Language - Download Page.
- In addition, you'll need the ROBOTC modified bootloader. You can download that here.
- ROBOTC Modified UNO Bootloader - Modified Bootloader.

Bootload Download Instructions:

- Download the Arduino Open Source Software and a copy of the Modified Bootloader File



Name	Date modified	Type	Size
avrinterruptnames.h	11/28/2011 6:32 PM	C/C++ Header	20 KB
command.h	11/28/2011 6:32 PM	C/C++ Header	5 KB
License.txt	11/28/2011 6:32 PM	Text Document	16 KB
Makefile	11/28/2011 6:32 PM	File	18 KB
stk500boot.c	11/28/2011 6:32 PM	C File	50 KB
stk500boot.ppg	11/28/2011 6:32 PM	PPG File	1 KB
stk500boot_v2_mega2560.hex	11/28/2011 6:32 PM	HEX File	23 KB
STK500V2.pnproj	11/28/2011 6:32 PM	PNPROJ File	1 KB
STK500V2.pnps	11/28/2011 6:32 PM	PNPS File	1 KB

Fig 4.11

- Power up your Arduino UNO (either via USB or external power)
- Plug in your AVR ISP Programmer to your computer (make sure you have any required drivers installed)
- Connect your AVR ISP Programmer into your Arduino UNO Board via the ISP Header (the 2x3 header pins right above the Arduino Logo)
- Launch the Arduino Open Source Software

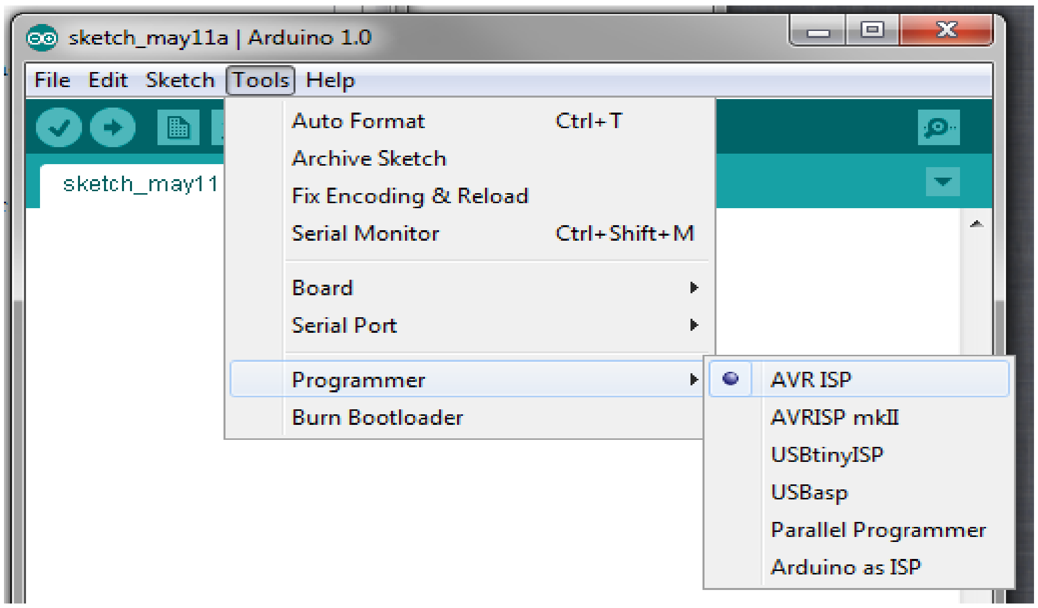


Fig 4.12

• Select the "Burn Bootloader" option under the "Tools" menu. The modified bootloader will now be sent to your Arduino. This typically take a minute or so.

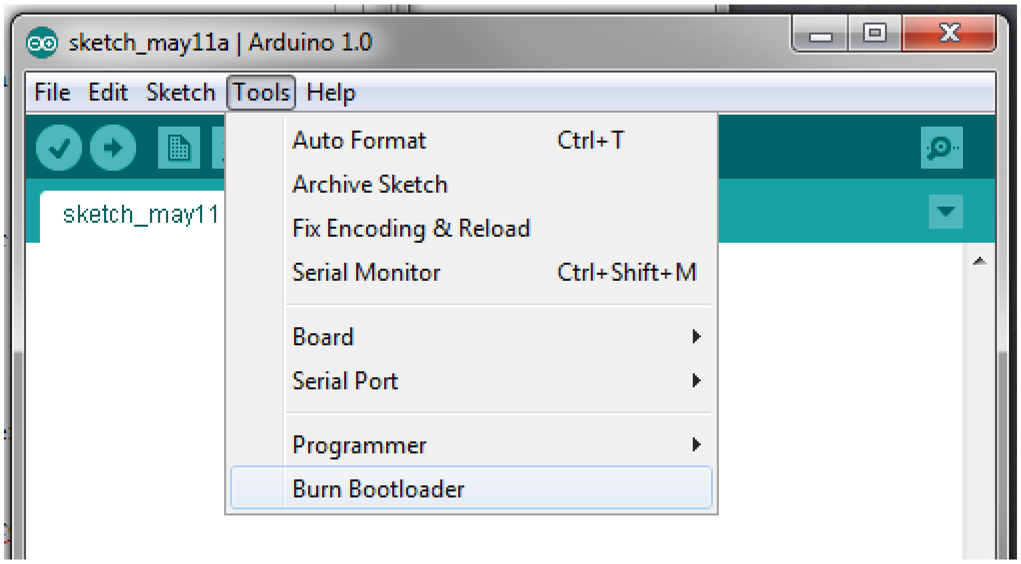


Fig 4.13

- You should be all set to download ROBOTC firmware and start using your Arduino UNO with ROBOTC

Technical Details:

The Arduino Boot loader sets the "erase Address" to zero every time the boot loader is called. ROBOTC called the "Load Address" command to set the address in which we want to write/verify when downloading program.

When writing a page of memory to the arduino, the Arduino boot loader will erase the existing page and write a whole new page.

In the scenario of downloading firmware, everything is great because the Erase Address and the Loaded Address both start at zero.

In the scenario of writing a user program, we start writing at memory location 0x7000, but the Boot loader erases information starting at location zero because the "Load Address" command doesn't update where to erase.

Our modification is to set both the Load Address and the Erase Address so the activity of writing a user program doesn't cause the firmware to be accidentally erased.

Summary:

Microcontroller	- Arduino UNO
Operating Voltage	InputVoltag
(recommended)Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40mA

DC Current for3.3VPin	50mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8KB
EEPROM	4KB
Lock Speed	16MHz

The power pins are as follows:

VIN:

- It is the input voltage to the Arduino board when using external power.
- supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V:

- Regulated 5V power supply used to power microcontroller.
- This can come either from VIN via a non-board regulator, or be supplied by USB or another regulated 5V supply.

3V3:

- 3volt supply generated by theon-boardregulator.
- Maximum current draw is 50mA

GND. Ground pins:

- The ATMEGA has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).
- They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50k Ohms.

Communication:

- The Arduino UNO has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The Arduino UNO provides four hardware UARTs for TTL (5V) serial communication.\
- An ATMEGA on the board channels one of these over USB and provides a virtual comport to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer.
- A Software Serial library allows for serial communication on any of the digital pins.
- The Arduino UNO also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the Arduino UNO datasheet.

Programming:

The Arduino UNO can be programmed with the Arduino software (download). For details, see the reference and tutorials**.**

The Arduino UNO on the Arduino UNO comes pre burned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header.

Automatic (Software) Reset :

- The Arduino UNO is designed to be reset by software running on a connected computer, allowing code to be uploaded by simply pressing the upload button. This allows the bootloader to have a shorter timeout.
- The Arduino UNO resets each time a connection is made to it from software, and the bootloader will intercept the first few bytes of data sent after a connection is opened.
- The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

4.6.2 EMBEDDED C:

High-level language programming has long been in use for embedded-systems development. However, assembly programming still prevails, particularly for digital-signal processor (DSP) based systems. DSPs are often programmed in assembly language by programmers who know the processor architecture inside out. The key motivation for this practice is performance, despite the disadvantages of assembly programming when compared to high-level language programming.

- DSP processors have a data path with memory-access units that directly feed into arithmetic units, and address registers are placed next to the memory units in a separate register file.
- Localization of resources in the data path saves data movements in a Load-Store architecture, resulting in a single cycle Multiply-accumulate unit (MAC).
- Fixed-point arithmetic is an important extension to DSP architectures, as it reduces the number of control-flow instructions needed for checking overflow. Changes in technological and economic requirements make it more expensive to continue programming DSPs in assembly.
- DSP programming is still done in assembly due to the lack of efficient and natural expression of algorithms in Standard C.
- Standard C does not provide a primitive for saturated arithmetic, requiring comparisons, conditional statements, and correcting assignments.

DESCRIPTION:

Embedded C is designed to bridge the performance mismatch between Standard C and the embedded hardware and application architecture. It extends the C language with the primitives that are needed by signal-processing applications and that are commonly provided by DSP processors. The design of the support for fixed-point data types and named address spaces in Embedded C is based on DSP-C. DSP-C [1] is an industry-designed extension of C with which experience was gained since 1998 by various DSP manufacturers in their compilers. For the development of DSP-C by ACE cooperation was sought with embedded-application designers and DSP manufacturers

The Embedded C specification extends the C language to support freestanding embedded processors in exploiting the multiple address space functionality, user-defined named address spaces, and direct access to processor and I/O registers. These features are common for the small, embedded processors used in most consumer products. The features introduced by Embedded C are fixed-point and saturated arithmetic, segmented memory spaces, and hardware I/O addressing. The description we present here addresses the extensions from a language-design perspective, as opposed to the programmer or processor architecture perspective.

MULTIPLE ADDRESS SPACES:

Embedded C supports the multiple address spaces found in most embedded systems. It provides a formal mechanism for C applications to directly access (or map onto) those individual processor instructions that are designed for optimal memory access. Named address spaces use a single, simple approach to grouping memory locations into functional groups to support MAC buffers in DSP applications, physical separate memory spaces, direct access to processor registers, and user-defined address spaces.

Embedded C uses address space qualifiers to identify specific memory spaces in variable declarations. There are no predefined keywords for this, as the actual memory segmentation is left to the implementation. As an example, assume that X and Y are memory qualifiers.

For proper integration with the C language, a memory structure is specified, where the unqualified memory encompasses all other memories. All unqualified pointers are pointers into this unqualified memory. The unqualified memory abstraction is needed to keep the compatibility of the void * type, the NULL pointer, and to avoid duplication of all library code that accesses memory through pointers that are passed as parameters.

NAMED REGISTERS:

Embedded C allows direct access to processor registers, declared and used like conventional C variables, allowing developers to develop applications in a high-level language. Named address spaces and processor access reduce application dependency on assembly code.

I/O HARDWARE ADDRESSING:

The motivation to include primitives for I/O hardware addressing in Embedded C is to improve the portability of device-driver code. In principle, a hardware device driver should only be concerned with the device itself. The driver operates on the device through device registers, which are device specific. The method to access these registers can be very different on different systems, even though it is the same device that is connected. The I/O hardware access primitives aim to create a layer that abstracts the system-specific access method from the device that is accessed. The ultimate goal is to allow source-code portability of device drivers between different systems. In the design of the I/O hardware-addressing interface, three requirements needed to be fulfilled.

- The device-drive source code must be portable.
- The interface must not prevent implementations from producing machine code that is as efficient as other methods.
- The design should permit encapsulation of the system-dependent access method.

EMBEDDED C PORTABILITY :

By design, a number of properties in Embedded C are left implementation defined. This implies that the portability of Embedded C programs is not always guaranteed. Embedded C provides access to the performance features of DSPs. As not all processors are equal, not all Embedded C implementations can be equal For example, suppose an application requires 24-bit fixed-point arithmetic and an Embedded C implementation provides only 16 bits because that is the native size of the processor. When the algorithm is expressed in Embedded C, it will not produce outputs of the right precision.

Embedded C provides a great improvement in portability and software engineering of embedded applications, despite differences in performance-specific processors.

The porting of an automotive engine controller application from a special-purpose processor to a general-purpose 8-bit Freescale 68S08 was easier than expected, and the performance of the special-purpose processor was significantly higher than the general-purpose target

CHAPTER 5

RESULT AND DISCUSSION

The complete setup is depicted in fig 5.1The setup consists of gsr sensor and lcd display for processing the stress level. Additionally, a buzzer to alert the patient.The gsr sensor send the result to the lcd display using the Wifi module

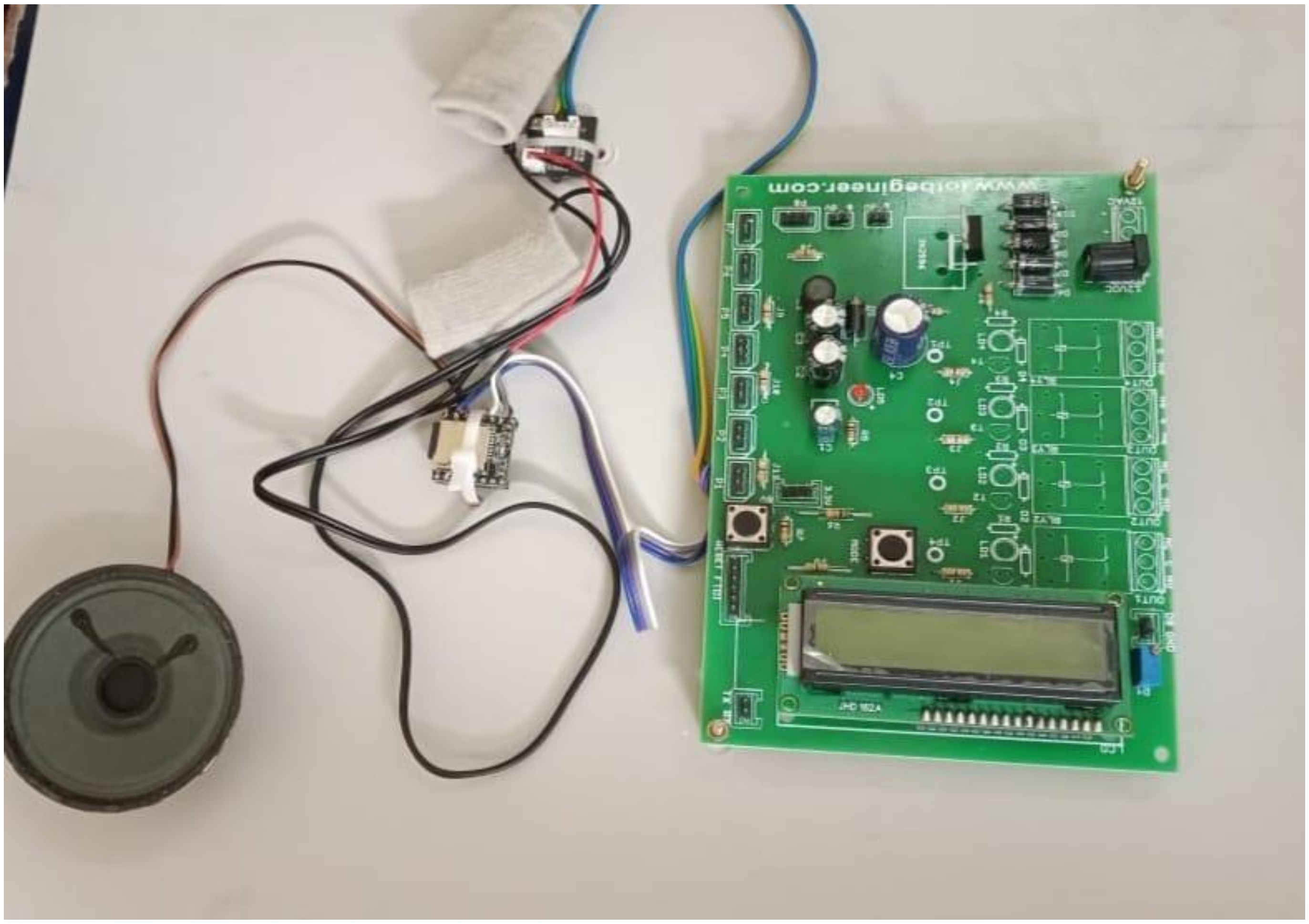
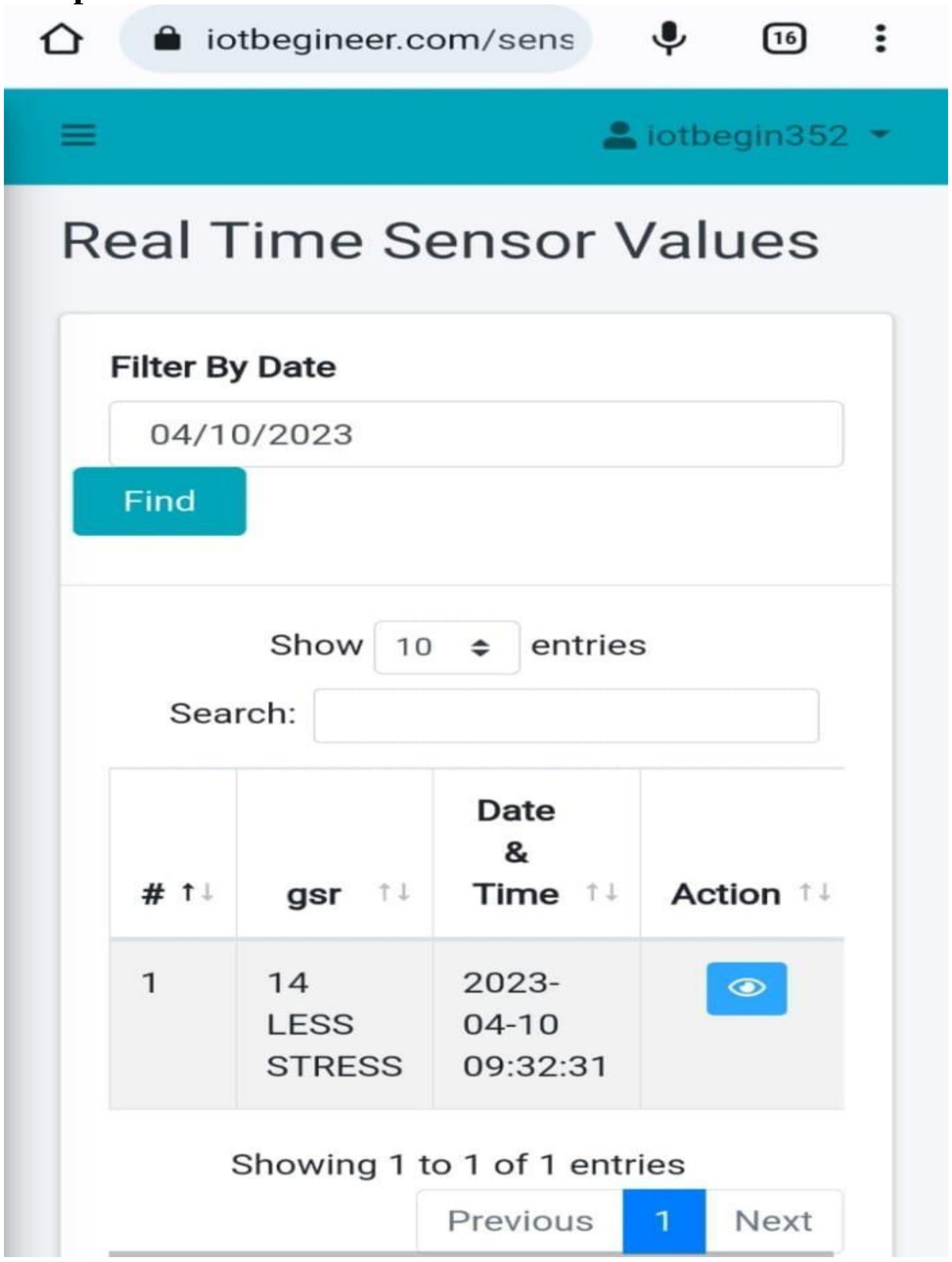


Fig 5.1 Complete setup of the system

Sample Data



iotbegineer.com/sens

iotbegin352

Real Time Sensor Values

Filter By Date

04/10/2023

Find

Show 10 entries

Search:

#	gsr	Date & Time	Action
1	14 LESS STRESS	2023-04-10 09:32:31	

Showing 1 to 1 of 1 entries

Previous 1 Next

Fig 5.2 Representation of IOT webpage

5.3 CONCLUSION:

With the wide use of internet, this work is concentrated to execute the internet technology to establish a system which would communicate through internet for better health. Internet of Things rules the whole world in various fields, mainly in health care sectors. Hence the present work is done to design an Internet of Things based smart stress monitoring system using microcontroller.

5.4 Future Work :

Overall, future work for real-time stress analysis using GSR sensors could involve the development of wearable devices, machine learning algorithms, contextual information integration, real-time feedback and intervention strategies, telehealth and digital health applications, long-term stress monitoring methods, and validation and standardization efforts. These advancements could potentially enable more accurate, personalized, and effective stress monitoring and management in real-time, with the goal of improving mental health and well-being.

APPENDIX

```
GSR connection pins to Arduino microcontroller

Arduino           GSR

GND               GND
5V                VCC
A2                SIG

D13             RED LED

*/

/*
 GSR, standing for galvanic skin response, is a method of
 measuring the electrical conductance of the skin. Strong
 emotion can cause stimulus to your sympathetic nervous
 system, resulting more sweat being secreted by the sweat
 glands. Grove – GSR allows you to spot such strong emotions
 by simple attaching two electrodes to two fingers on one
hand,
 an interesting gear to create emotion related projects, like
 sleep quality monitor.
http://www.seeedstudio.com/wiki/Grove_-_GSR_Sensor
 */

const int GSR=36;
int  threshold=0;
int sensorValue;
#include "Arduino.h"
#include "SoftwareSerial.h"
#include "DFRobotDFPlayerMini.h"

//SoftwareSerial mySoftwareSerial(16, 17); // RX, TX
DFRobotDFPlayerMini myDFPlayer;
void printDetail(uint8_t type, int value);
#include <HTTPClient.h>
#include <WiFi.h>
```

```
#include <ArduinoJson.h>

#include<LiquidCrystal.h>
const int rs = 5, en = 18, d4 = 19, d5 = 21, d6 = 22, d7 =
23;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

String        sensors;
String sensor1_status;
String sensor2_status;
String sensor3_status;
String sensor4_status;
String sensor5_status;
String sensor6_status;
String sensor7_status;
String sensor8_status;
String sms_status;
int stress;
void setup()
{
  long sum=0;
  Serial2.begin(9600);
  Serial.begin(9600);
  lcd.begin(16,2);
   WiFi.begin("iotbegin352", "iotbegin352");   //WiFi
connection

    while (WiFi.status() != WL_CONNECTED)
    {  //Wait for the WiFI connection completion
//      blink_led();
lcd.setCursor(0,0);
lcd.print("WIFI Connect to");
lcd.setCursor(4,1);
lcd.print("iotbegin352");
   //   Serial.println("Waiting for Wi-Fi connection");
    }

  Serial.println();
  Serial.println(F("DFRobot DFPlayer Mini Demo"));
  Serial.println(F("Initializing DFPlayer ... (May take 3~5
seconds)"));
```

```
  if (!myDFPlayer.begin(Serial2)) {  //Use softwareSerial to
communicate with mp3.
    Serial.println(F("Unable to begin:"));
    Serial.println(F("1.Please recheck the connection!"));
    Serial.println(F("2.Please insert the SD card!"));
   // while(true){
   //   delay(0); // Code to compatible with ESP8266 watch
dog.
    //}
  }
  Serial.println(F("DFPlayer Mini online."));

  myDFPlayer.volume(30);

  delay(1000);

  for(int i=0;i<500;i++)
  {
  sensorValue=analogRead(GSR);
  sum += sensorValue;
  delay(5);
  }
  threshold = sum/500;
   Serial.print("threshold =");
   Serial.println(threshold);
   lcd.clear();
  }

void loop(){
  int temp;
  sensorValue=analogRead(GSR);
  Serial.print("sensorValue=");
  Serial.println(sensorValue);
  temp = threshold - sensorValue;
  if(abs(temp)>60)
  {
    sensorValue=analogRead(GSR);
    temp = threshold - sensorValue;
    if(abs(temp)>60){
    Serial.println("Emotion Changes Detected!");
    delay(3000);
    delay(1000);
   }
  }

stress =  (4095 - sensorValue ) / 40;
lcd.setCursor(0,0);
lcd.print("STRESS LEVEL:");
lcd.print(stress);
lcd.print("  ");
if ((stress < 50)&&(stress > 0))
{
lcd.setCursor(0,1);
lcd.print("LESS STRESS");
myDFPlayer.play(1);
 sensor1_status = String(stress) + " LESS STRESS" ;
 iot();
}
else if ((stress < 75)&&(stress > 50))
{
lcd.setCursor(0,1);
lcd.print("MEDIUM STRESS");
myDFPlayer.play(2);
 sensor1_status = String(stress) + " MEDIUM STRESS" ;
 iot();
}
else if ((stress < 100)&&(stress > 75))
{
  lcd.setCursor(0,1);
lcd.print("HIGH STRESS");
myDFPlayer.play(3);
 sensor1_status = String(stress) + " HIGH STRESS" ;
 iot();
}
else
{
  lcd.setCursor(0,1);
lcd.print("                     ");
}
```

```
delay(1000);


}


void iot()
{

DynamicJsonDocument jsonBuffer(JSON_OBJECT_SIZE(3) + 300);
JsonObject root = jsonBuffer.to<JsonObject>();

root["sensor1"] = sensor1_status;
root["sensor2"] = sensor2_status;
root["sensor3"] = sensor3_status;
root["sensor4"] = sensor4_status;
root["sensor5"] = sensor5_status;
root["sensor6"] = sensor6_status;
root["sensor7"] = sensor7_status;
root["sensor8"] = sensor8_status;
root["sms"]     = sms_status;
String json;
serializeJson(jsonBuffer, json);
if (sensor1_status != "null")
{
HTTPClient http;    //Declare object of class HTTPClient
http.begin("http://iotbegineer.com/api/sensors");
//Specify request destination
http.addHeader("username",  "iotbegin352"); //Specify
content-type header
http.addHeader("Content-Type",  "application/json");
int httpCode = http.POST(json);    //Send the request
String payload = http.getString();//Get the response
payload
http.end(); //Close connection
delay(5000);

}

}
```

REFERENCES

[1] J. G. Rodrigues, M. Kaiseler, A. Aguiar, J. P. S. Cunha, and J. Barros, "A mobile sensing approach to stress detection and memory activation for public bus drivers," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 6, pp. 3294–3303, 2015.

[2] A. Vahedian-Azimi, M. Hajiesmaeili, M. Kangasniemi, J. Fornés-Vives, R. L. Hunsucker, F. Rahimibashar, M. A. Pourhoseingholi, L. Farrokhvar, and A. C. Miller, "Effects of stress on critical care nurses: a national cross-sectional study," Journal of intensive care medicine, vol. 34, no. 4, pp. 311–322, 2019.

[3] C. Epp, M. Lippold, and R. L. Mandryk, "Identifying emotional states using keystroke dynamics," in Proceedings of the sigchi conference on human factors in computing systems. ACM, 2011, pp. 715–724.

[4] T. S. Saini and M. Bedekar, "Inferring user emotions from keyboard and mouse," in Intelligent Computing and Information and Communication. Springer, 2018, pp. 591–601.

[5] A. Kołakowska, "Recognizing emotions on the basis of keystroke dynamics," in 2015 8th International Conference on Human System Interaction (HSI). IEEE, 2015, pp. 291–297.

[6] H.-R. Lv, Z.-L.Lin, W.-J.Yin, and J. Dong, "Emotion recognition based on pressure sensor keyboards," in 2008 IEEE International Conference on Multimedia and Expo. IEEE, 2008, pp. 1089–109.

[7]T. Morris, P. Blenkhorn and F. Zaidi. "Blink detection for real-time eye tracking". Journal of Network and Computer Applications, Vol. 25, num. 2, pp. 129-143, 2002.

[8]Abdallah A. Alshennawy, and Ayman A. Aly, "Edge Detection in Digital Images Using Fuzzy Logic Technique", World Academy of Science, Engineering and Technology 51, 2009.

[9] Lim, S.,Lee, K., Byeon, O., Kim, T, "Efficient Iris Recognition through Improvement of Feature Vector and Classifier", ETRJ Journal, Vol. 23, Number 2, pp. 61-70, 2001.

[10] Inoue T, Abe S. Fuzzy support vector machines for pattern classification. In: Proceeding of International Joint Conference on Neural Networks. Washington DC, pp.1449–1454, 2001.

[11]N.Ozlem Ozcan, Fikret Gurgen, “Fuzzy Support Vector Machines for ECG Arrhythmia Detection”, International Conference on Pattern Recognition, pp.2973-2976,2010.

[12]J ing Zhai, Armando Barreto: Stress Recognition Using Non-invasive Technology. FLAIRS Conference, pp. 395-401, 2006.