

INTELLIGENT BABY CRADLE

submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering in Electronics and Communication Engineering

BY

PARIMALA D (38130154)

NAJMUS SEHAR (38130146)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MARCH – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

www.sathyabama.ac.in



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **PARIMALA D (38130154)** and **NAJMUS SEHAR (38130146)** who carried out the project entitled “**INTELLIGENT BABY CRADDLE**” under our supervision from December 2021 to May 2022.

Internal Guide

Dr. T. GOMATHI, M.Tech, Ph.D.,

Head of the Department

Dr. T. RAVI, M.E., Ph.D.,

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

We **PARIMALA D (38130154)** and **NAJMUS SEHAR (38130146)** hereby declare that the Project Report entitled “**INTELLIGENT BABY CRADLE**” done by us under the guidance of **Ms. T. GOMATHI, M.Tech., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering in Electronics and Communication.

DATE:

1. PARIMALA D (38130154)

2. NAJMUS SEHAR (38130146)

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

We convey our thanks to **Dr. N.M. NANDITHA, M.E., Ph.D.**, Dean, School of Electrical and Electronics Engineering and **Dr. T. RAVI, M.E., Ph.D.**, Head of the Department, Dept. of Electronics and Communication Engineering for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Ms.T. GOMATHI, M.Tech., (Ph.D.)** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

Since we are in the twenty first century parents of small child are do jobs or work. They might have work to be done at home also. So simultaneously doing work and monitoring the baby could be trouble.

So, for these many kinds of parent we are developing a cradle which consist of too many useful features. Cradle system has ability to send alert when baby is crying, it can also swing automatically when baby is crying.

Cradle system also gives alert when baby does potty in the cradle. Cradle also has the ability to send alert regarding body temperature of the baby.

There is some High alert also available regarding babies' health. Likewise, there are multiple features in the cradle system. For sending the alert we are using GSM module. Cradle system is the solution of the multiple problems.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF FIGURES	vii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
3	AIM AND SCOPE OF THE PRESENT INVESTIGATION	4
	3.1 AIM	4
	3.2 EXISTING SYSTEM	4
	3.3 PROPOSED SYSTEM	4
	3.4 BLOCK DIAGRAM	5
	3.5 REQUIRED COMPONENTS	5
4	ALGORITHMS USED	6
	4.1 INTERNET OF THINGS (IoT)	6
	4.2 SOFTWARE REQUIRED	49
	4.3 UBIDOTS EXPLORER	61
5	RESULT AND DISCUSSION	71
6	SUMMARY AND CONCLUSION	75
	REFERENCES	76

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	Block Diagram	5
4.1	Internet of Things	6
4.2	Example of IoT	7
4.3	IoT Architecture	7
4.4	Basic Embedded System	10
4.5	Typical IoT ecosystem	11
4.6	The IoT Technology Stack	12
4.7	The IoT Technology Stack	13
4.8	Components of IoT Architecture	14
4.9	The 4 stage IoT Solution Architecture	16
4.10	Digital Devices in our daily life	17
4.11	Raspberry Pi	18
4.12	Arduino	18
4.13	ESP8266	19
4.14	Sense HAT	19
4.15	Pin Diagram	20
4.16	IoT Platform	21
4.17	NodeMCU	24
4.18	NodeMCU Development Board/kit v0.9	24
4.19	Image source – Seed Studio	26
4.20	ESP32-CAM	27
4.21	Image source – Seed Studio	28
4.22	AI-THINKER Model	30
4.23	ESP32-CAM	31
4.24	Interfacing DHT11 sensor with PIC16F628A	35
4.25	"Start" and "Response" signals	36
4.26	Start, Response and Data signals in sequence	37
4.27	Circuit connections for PIC16F628A and DHT11 sensor	38
4.28	18-pin PIC16F board	39

FIGURE NO	TITLE	PAGE NO
4.29	Simple sensors for gardeners	41
4.30	Gas sensor module	41
4.31	Sound sensor consist of Microphone	43
4.32	Arduino Uno	44
4.33	GIF shows that the LED is ON, as the clap	46
4.34	Small R/C servo mechanism	48
4.35	Preferences	57
4.36	Ubidots Data Hierarchy	61
4.37	Ubidots time series	63

LIST OF ABBREVIATIONS

AWS	- Amazon Web Services
BEC	- Battery Eliminator Circuit
ESC	- Electronic Speed Controller
IDE	- Integrated Development Environment
IoT	- Internet of things
LAN	- Local Area Network
PPM	- Pulse Position Modulation
USB	- Universal Serial Bus
WAN	- Wide Area Network

CHAPTER 1

INTRODUCTION

The Internet of things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interactions.

The Internet of Things thus promises a strong added value to each domain. By connecting objects, people and environments, it becomes possible to develop improvements that can only be beneficial. As we all know the term IOT that is internet of things which is designed to save time and for making work easy and accurate. As far as time and security is concern with the help of IOT we will build a cradle system which will make parents stress free and most important it will be safe and secure for the baby.

So, managing the work in time and taking care of baby is very important factors. Cradle system will give parents required time to parent for rest, as if the parents both mother and father goes for the job or even if the mother is house wife. Being stress free will definitely create the great atmosphere which will make great atmosphere around the baby.

So, it doesn't matter if there is no one to swing cradle it will do swing automatically if the baby is crying. It doesn't matter if baby has done pee and no one knows about for long time, but not need to worry cradle system will also give the alert about the wetness in cradle. Also, if baby is getting fever or cold Cradle system also have the ability to detect it and send the alert. Our cradle will also able to detect if any motion in the cradle by motion sensor for security purpose of the baby. Proposed system will help the parents, so that they can take good care of their baby.

CHAPTER 2

LITERATURE SURVEY

Harshad Suresh Gare, Bhushan Kiran Shahne, Kavita Suresh Jori³, Sweety G. Jachak, IOT Based Smart Cradle System for Baby Monitoring.

In the digital world of modern parents, duty of caring new born baby mostly on women which plays role of hurdle and also question marked on secureness, health, comfort, etc. of baby. So, for these people who belief in technology for them we are proposing automated cradle system which will connected to the parent's mobile for sending alert message.

S. Kavitha; R.R. Neela; M. Sowndarya; Madhuchandra; K. Harshitha, Analysis on IoT Based Smart Cradle System with an Android Application for Baby Monitoring

A system of interrelated computing devices, mechanical, and digital machines that are provided with the ability to transfer data over a network without requiring human interaction constitutes Internet of Things. This brings out automation of things. It is achieved through sensor and actuator devices. This paper brings out a survey on various sensors and actuator which is used in the implementation of Smart Cradle.

Prof. A.B. Tupkar, Prajwal. Chahare, Shubham. Rade, Rushikesh. Wakade, Snehal. Bahirseth, Development of IoT Based Smart Baby Cradle

The cornerstone of our project is Women Empowerment; by providing them with an Automatic Baby Monitoring System, with a special focus on developing countries. This project is the personification of a Smart Baby Cradle, brought about by integrating distinctive features i.e. automatic cradle swing (using cry detection system), interactive toys and communication module (for monitoring purposes), in a

single unit. The current era of digitization provides a large-scale availability of data as well as computing capability which can be used to bridge the gap between a child and a working mother.

Waheb A. Jabbar, Hiew Kuet Shang, Saidatul N. I. S. Hamid, Akram A. Almohammed, Roshahliza M. Ramli, and Mohammed A. H. Ali, IoT-BBMS: Internet of Things-Based Baby Monitoring System for Smart Cradle

The current number of working mothers has greatly increased. Subsequently, baby care has become a daily challenge for many families. Thus, most parents send their babies to their grandparents' house or to baby care houses. However, the parents cannot continuously monitor their babies' conditions either in normal or abnormal situations.

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 AIM

Proposed framework targets observing the indispensable indications of the infant, for example, internal heat level utilizing remote innovation and sound sensor. We likewise center around increment the extent of sending the data over the web to give far off access. The camera module fused empowers showing the visual feeds of the whereabouts of the infant and keeping an eye over their developments in a limited territory. This framework beats the downside of the current frameworks which are ungainly, less easy to understand and costly.

3.2 EXISTING SYSTEM

In existing system there no smart system for monitoring the children, parents should be there to care of the children's. So now a days it is so difficult to monitoring the baby continuously because in the growing technology there is no time for see all work simultaneously.

3.3 PROPOSED SYSTEM

Many of IOT devices are being develop in the IT sector. There are some cradles also, which are built with integration of IOT, but still there are some less feature which could be threat to the health of the babies. As we have seen in India or any other industrializing nation that both parents need to go to work and also look after the baby which increases workload on both the parent, it could also affect their professional life and their babies' life. Due to less featured cradle systems and parents busy schedule we are implementing modern day cradle system.

3.4 BLOCK DIAGRAM

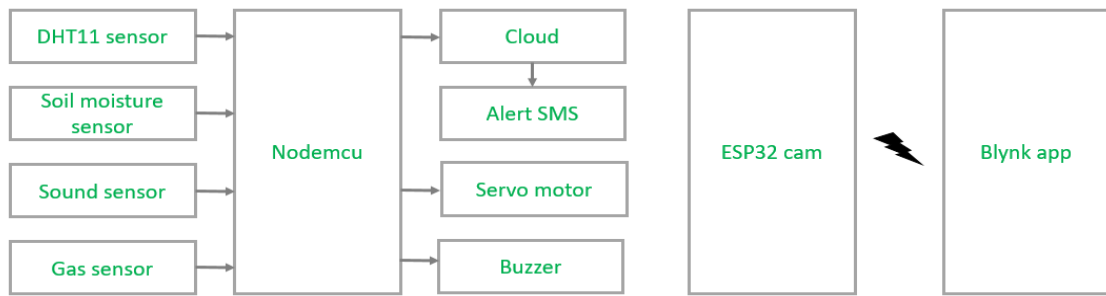


Fig. 3.1: Block Diagram

3.5 REQUIRED COMPONENTS

Hardware used

- Nodemcu
- DHT11 sensor
- Soil moisture sensor
- Sound sensor
- Gas sensor
- ESP32 CAM

Software used

- Arduino Ide
- Ubidots explorer

CHAPTER 4

ALGORITHMS USED

IoT stands for Internet of Things, which means accessing and controlling daily usable equipments and devices using Internet.

Our IoT tutorial includes all topics of IoT such as introduction, features, advantage and disadvantage, ecosystem, decision framework, architecture and domains, biometric, security camera and door unlock system, devices, etc.

4.1 INTERNET OF THINGS (IoT)

Let's us look closely at our mobile device which contains GPS Tracking, Mobile Gyroscope, Adaptive brightness, Voice detection, Face detection etc. These components have their own individual features, but what about if these all communicate with each other to provide a better environment? For example, the phone brightness is adjusted based on my GPS location or my direction.

The term "Things" in the Internet of Things refers to anything and everything in day to day life which is accessed or connected through the internet.

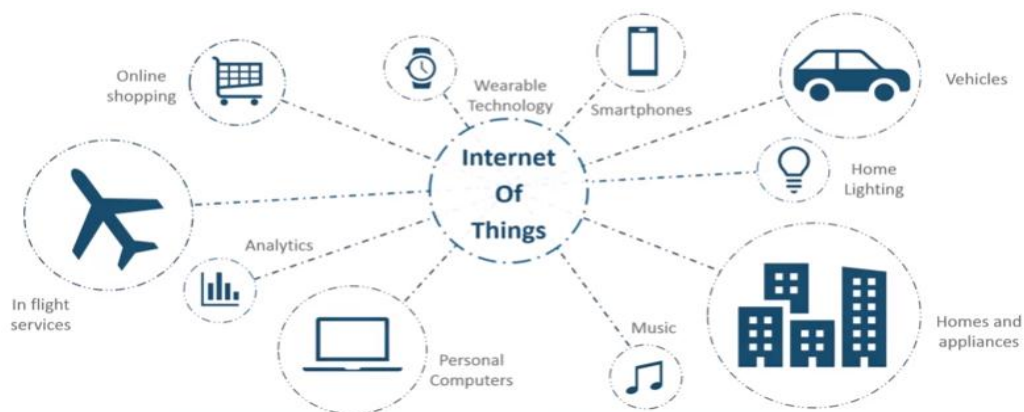


Fig. 4.1: Internet of Things

IoT is an advanced automation and analytics system which deals with artificial intelligence, sensor, networking, electronic, cloud messaging etc. to deliver complete systems for the product or services. The system created by IoT has greater transparency, control, and performance.

As we have a platform such as a cloud that contains all the data through which we connect all the things around us. For example, a house, where we can connect our home appliances such as air conditioner, light, etc. through each other and all these things are managed at the same platform. Since we have a platform, we can connect our car, track its fuel meter, speed level, and also track the location of the car.



Fig. 4.2: Example of IoT

How does Internet of Thing (IoT) Work?

The working of IoT is different for different IoT echo system (architecture). However, the key concept of there working are similar. The entire working process of IoT starts with the device themselves, such as smartphones, digital watches, electronic appliances, which securely communicate with the IoT platform. The platforms collect and analyze the data from all multiple devices and platforms and transfer the most valuable data with applications to devices.

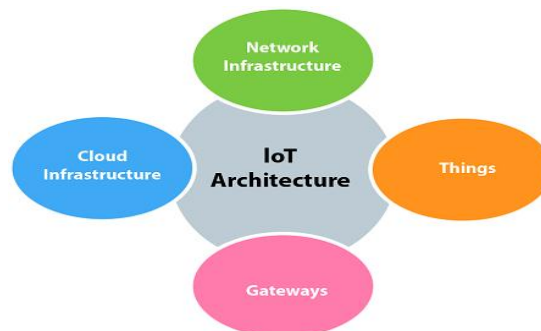


Fig. 4.3: IoT Architecture

Features of IOT

The most important features of IoT on which it works are connectivity, analyzing, integrating, active engagement, and many more. Some of them are listed below:

Connectivity: Connectivity refers to establish a proper connection between all the things of IoT to IoT platform it may be server or cloud. After connecting the IoT devices, it needs a high speed messaging between the devices and cloud to enable reliable, secure and bi-directional communication.

Analyzing: After connecting all the relevant things, it comes to real-time analyzing the data collected and use them to build effective business intelligence. If we have a good insight into data gathered from all these things, then we call our system has a smart system.

Integrating: IoT integrating the various models to improve the user experience as well.

Artificial Intelligence: IoT makes things smart and enhances life through the use of data. For example, if we have a coffee machine whose beans have going to end, then the coffee machine itself order the coffee beans of your choice from the retailer.

Sensing: The sensor devices used in IoT technologies detect and measure any change in the environment and report on their status. IoT technology brings passive networks to active networks. Without sensors, there could not hold an effective or true IoT environment.

Active Engagement: IoT makes the connected technology, product, or services to active engagement between each other.

Endpoint Management: It is important to be the endpoint management of all the IoT system otherwise, it makes the complete failure of the system. For example, if a coffee machine itself order the coffee beans when it goes to end but

what happens when it orders the beans from a retailer and we are not present at home for a few days, it leads to the failure of the IoT system. So, there must be a need for endpoint management.

Advantages and Disadvantages of (IoT)

Any technology available today has not reached to its 100 % capability. It always has a gap to go. So, we can say that **Internet of Things** has a significant technology in a world that can help other technologies to reach its accurate and complete 100 % capability as well.

Let's take a look over the major, advantages, and disadvantages of the Internet of Things.

Advantages of IoT

Internet of things facilitates the several advantages in day-to-day life in the business sector. Some of its benefits are given below:

- ***Efficient resource utilization:*** If we know the functionality and the way that how each device work we definitely increase the efficient resource utilization as well as monitor natural resources.
- ***Minimize human effort:*** As the devices of IoT interact and communicate with each other and do lot of task for us, then they minimize the human effort.
- ***Save time:*** As it reduces the human effort then it definitely saves out time. Time is the primary factor which can save through IoT platform.
- ***Enhance Data Collection:***
- ***Improve security:*** Now, if we have a system that all these things are interconnected then we can make the system more secure and efficient.

Disadvantages of IoT

As the Internet of things facilitates a set of benefits, it also creates a significant set of challenges. Some of the IoT challenges are given below:

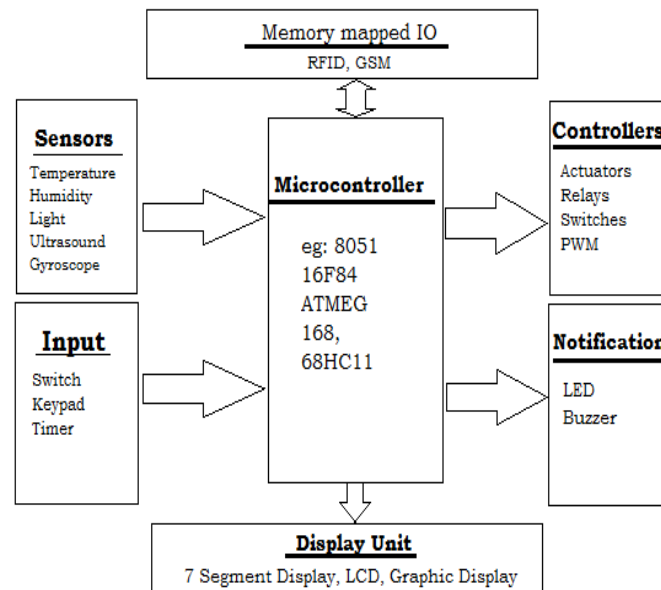


Fig. 4:4 Basic Embedded System

IoT Ecosystem

The IoT ecosystem is not easy to define. It is also difficult to capture its proper image due to the vastness and emerging possibility and the rapidity with which it is expanding in the entire sector. However, the IoT ecosystem is a connection of various kind of devices that sense and analyze the data and communicates with each other over the networks.

The typical IoT ecosystem is shown in below image, where the smarter devices send and receive data from the devices themselves in the environment that are integrate over network and Cloud Computing.



Fig. 4.5: Typical IoT ecosystem

The IoT is itself an ecosystem of network devices that transfer the data. It is also well interconnected with Big Data and Cloud Computing.

- **Sensing, Embedded processing, Connectivity:** The IoT ecosystem senses its surrounding like temperature, gyroscope, pressure, etc. and make the embedded processing using devices. These devices are connected through any type of devices such as GPS, WiFi, RFID, etc. over the networks.
- **Smart devices and environment, Cloud Computing, Big Data:** The data transfer or receive through smart devices and environments are communicated through Cloud Computing or others Servers and stored as Big Data.
- **Technology, Software, Application:** The IoT ecosystem uses any of different technologies, software and application to communicate and connect with smart devices and environment.
- **Users or groups of community:** The product or services generated by the IoT ecosystem are consumed by the users or the group of communities to serve the smart life.

IoT Decision Framework

The IoT decision framework provides a structured approach to create a powerful IoT product strategy. The IoT decision framework is all about the strategic decision making. The IoT Decision Framework helps us to understand the areas where we need to make decisions and ensures consistency across all of our strategic business decision, technical and more.

The IoT decision framework is much more important as the product or services communicates over networks goes through five different layers of complexity of technology.

1. Device Hardware
2. Device Software
3. Communications
4. Cloud Platform
5. Cloud Application



Fig. 4.6: The IoT Technology Stack

Decision Area

The IoT decision framework pays attention to six key decision areas in any IoT product. These decision areas are:

1. Data
2. Business
3. Technology
4. Security
5. Standards & Regulations

Each of these decision areas is evaluated at each of the IoT Technology Stack. The User Experience will be evaluated at Device Hardware, Device Software and so to provide the better user experience. Then at the next step Data Decision Area, we have to explore data considerations for all the stages of IoT Technology Stack.

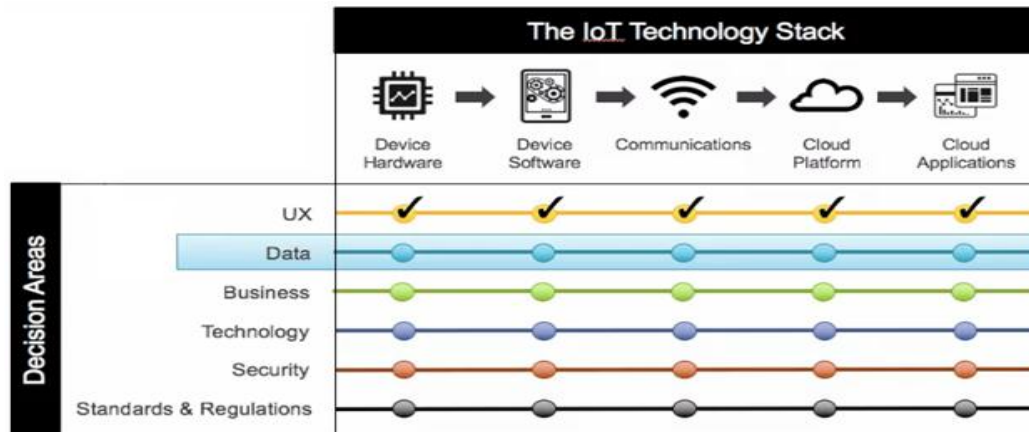


Fig. 4.7: The IoT Technology Stack

Decision Area of the IoT Decision Framework

Let's see each of the Decision Area of IoT Decision Framework in detail:

1. **User Experience Decision Area:** This is the area where we concentrate about who are the users, what are their requirements and how to provide a great experience at each step of IoT stack without worrying about the technical details.
2. **Data Decision Area:** In this area, we make the overall data strategy such as the data flow over the entire IoT stack to fulfill the user's requirements.
3. **Business Decision Area:** Based on the previous decisions area, we make the decision how product or services will become financial potential. At each of the IoT Stack level are monetized about the costs of providing services.

4. **Technology Decision Area:** In this area, we work with the technology for each layer to facilitate the final solution.
5. **Security Decision Area:** After going through the implementation of technology it is important to decide and provide the security at each stage of the IoT Stack.
6. **Standards & Regulations Decision Area:** At the last stage of IoT Decision Area, we identify the standards and regulations of product or services that will affect your product at each layer of the IoT Stack.

IoT Architecture

There is not such a unique or standard consensus on the Internet of Things (IoT) architecture which is universally defined. The IoT architecture differs from their functional area and their solutions. However, the IoT architecture technology mainly consists of four major components:

Components of IoT Architecture

- Sensors/Devices
- Gateways and Networks
- Cloud/Management Service Layer
- Application Layer

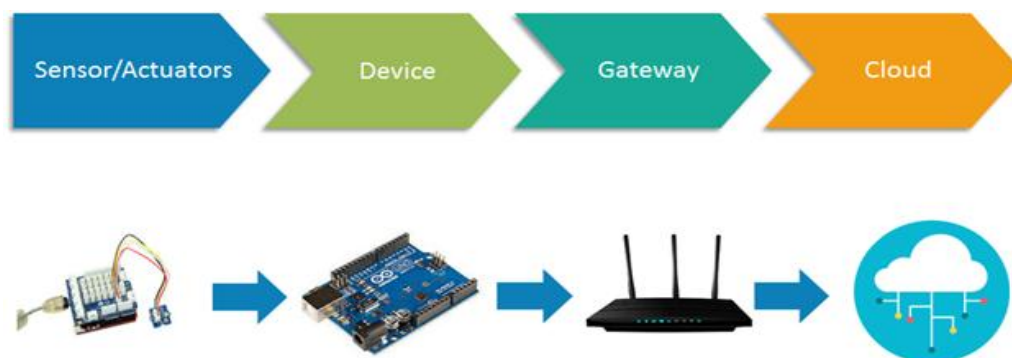


Fig. 4.8: Components of IoT Architecture

Stages of IoT Solutions Architecture

There are several layers of IoT built upon the capability and performance of IoT elements that provides the optimal solution to the business enterprises and end-users. The IoT architecture is a fundamental way to design the various elements of IoT, so that it can deliver services over the networks and serve the needs for the future.

Following are the primary stages (layers) of IoT that provides the solution for IoT architecture.

1. ***Sensors/Actuators:*** Sensors or Actuators are the devices that are able to emit, accept and process data over the network. These sensors or actuators may be connected either through wired or wireless. This contains GPS, Electrochemical, Gyroscope, RFID, etc. Most of the sensors need connectivity through sensors gateways
2. ***Gateways and Data Acquisition:*** As the large numbers of data are produced by this sensors and actuators need the high-speed Gateways and Networks to transfer the data. This network can be of type Local Area Network (LAN such as WiFi, Ethernet, etc.), Wide Area Network (WAN such as GSM, 5G, etc.).
3. ***Edge IT:*** Edge in the IoT Architecture is the hardware and software gateways that analyze and pre-process the data before transferring it to the cloud. If the data read from the sensors and gateways are not changed from its previous reading value then it does not transfer over the cloud, this saves the data used.
4. ***Data center / Cloud:*** The Data Center or Cloud comes under the Management Services which process the information through analytics, management of device and security controls. Beside this security controls and device management the cloud transfer the data to the end users application such as Retail, Healthcare, Emergency, Environment, and Energy, etc.

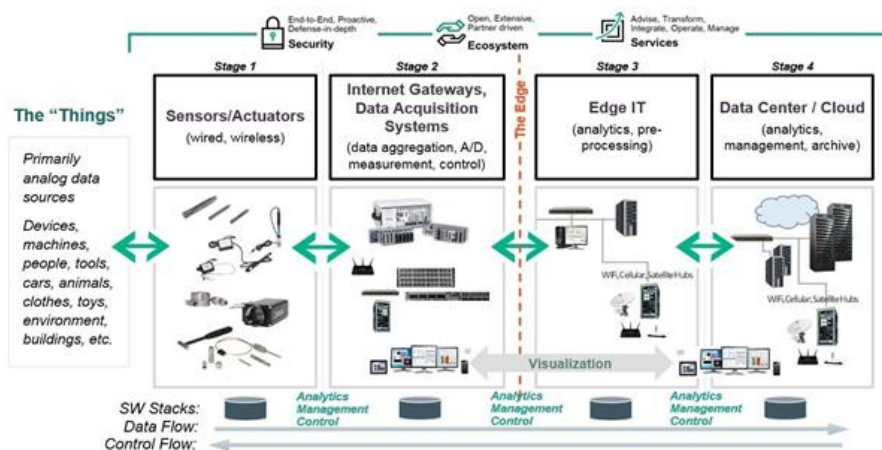


Fig. 4.9: The 4 stage IoT Solution Architecture

What are Smarts Objects in IoT

The concept of smart in IoT is used for physical objects that are active, digital, networked, can operate to some extent autonomously, reconfigurable and has local control of the resources. The smart objects need energy, data storage, etc.

In a day to day life, people have a lot of object with internet or wireless or wired connection. Such as:

- Smartphone
- Tablets
- TV computer

These objects can be interconnected among them and facilitate our daily life (smart home, smart cities) no matter the situation, localization, accessibility to a sensor, size, scenario or the risk of danger.



Fig. 4.10: Digital Devices in our daily life

Smart objects are utilized widely to transform the physical environment around us to a digital world using the Internet of things (IoT) technologies.

A smart object carries blocks of application logic that make sense for their local situation and interact with human users. A smart object sense, log, and interpret the occurrence within themselves and the environment, and intercommunicate with each other and exchange information with people.

Major IoT Boards in Market

There are several IoT boards available in the market to build the project. Some of the major IoT Boards are described below:

Raspberry Pi

Raspberry Pi is a much popular device used in building IoT project. The recently launched Raspberry Pi 3 includes built-in WiFi and Bluetooth making the most compact and standalone computer. It provides a powerful environment to install a variety of programming packages such as Python, Node.js, LAMP stack, Java and much more. Using 40 GPIO pins, and four USB ports you can connect many peripherals and accessories to the Pi.



Fig. 4.11: Raspberry Pi

Arduino

Arduino boards are the microcontrollers and microcontroller kit for building digital devices that can be sense and control objects in the physical and digital world. Arduino boards are furnished with a set of digital and analog input/output pins that may be interfaced to various other circuits. Some Arduino boards include USB (Universal Serial Bus) to load programs from the personal computer.



Fig. 4.12: Arduino

ESP8266

The ESP8266 is a low-cost Wi-Fi microchip with 32-bit microcontroller capability, standard digital peripheral interfaces. There are different types of ESP8266 boards available for different needs. The primary goal of this board is to deal with the built-in Wifi through AT commands if used as device module, but you can 'program' using Arduino board however it also reads and controls input/output, digital and analog.



Fig. 4.13: ESP8266

Sense HAT 8x8 RGB LED matrix

Raspberry Pi Sense HAT is an integrated sensor that can measure humidity, temperature, acceleration, and pressure. The 8x8 LED matrix display data read from Raspberry Pi Sense HAT sensors. The Sense HAT has an 8×8 RGB LED matrix includes the following sensors:

1. Gyroscope
2. Accelerometer
3. Magnetometer
4. Temperature
5. Barometric pressure
6. Humidity

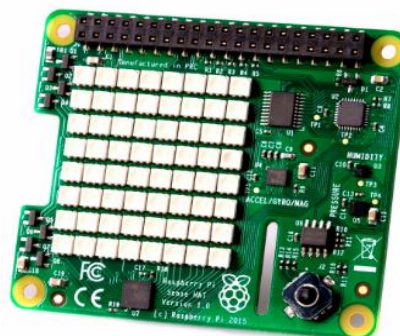


Fig. 4.14: Sense HAT

Bluetooth Module HC-05

Bluetooth Module HC-05 device is a 6 pins Bluetooth device that is used for wireless communication. Usually, this device connects small devices like mobile phones, PDAs and TVs using a short-range wireless connection to exchange data. It communicates with the microcontroller using the serial port (USART).

Pin Description

- **EN:** It is the enable pin, when it is connected to 3.3V then model is enabled.
- **+5V:** This is the supply pin for connecting +5V.
- **GND:** It is the ground pin.
- **TX:** It is the transmitter pin of the UART communication.
- **RX:** It is the receiver pin of UART communication.
- **STATE:** It indicates whether the module is connected or not. It acts as a status indicator.



Fig. 4.15: Pin Diagram

IoT - Platform

As in IoT, all the IoT devices are connected to other IoT devices and application to transmit and receive information using protocols. There is a gap between the IoT device and IoT application. An IoT Platform fills the gap between the devices (sensors) and application (network). Thus we can say that an IoT

platform is an integrated service that fulfills the gap between the IoT device and application and offers you to bring physical object online.



Fig. 4.16: IoT Platform

There are several IoT Platforms available that provides facility to deploy IoT application actively. Some of them are listed below:

Amazon Web Services (AWS) IoT platform: Amazon Web Service IoT platform offers a set of services that connect to several devices and maintain the security as well. This platform collects data from connected devices and performs real-time actions.

Microsoft Azure IoT platform: Microsoft Azure IoT platform offers strong security mechanism, scalability and easy integration with systems. It uses standard protocols that support bi-directional communication between connected devices and platform. Azure IoT platform has an Azure Stream Analytics that processes a large amount of information in real-time generated by sensors. Some common features provided by this platform are:

- Information monitoring
- A rules engine
- Device shadowing
- Identity registry

Google Cloud Platform IoT: Google Cloud Platform is a global cloud platform that provides a solution for IoT devices and applications. It handles a large amount of data using Cloud IoT Core by connecting various devices. It allows to apply BigQuery analysis or to apply Machine learning on this data. Some of the features provided by Google Cloud IoT Platform are:

- Cloud IoT Core
- Speed up IoT devices
- Cloud publisher-subscriber
- Cloud Machine Learning Engine

IBM Watson IoT platform: The IBM Watson IoT platform enables the developer to deploy the application and building IoT solutions quickly. This platform provides the following services:

- Real-time data exchange
- Device management
- Secure Communication
- Data sensor and weather data services

Artik Cloud IoT platform: Arthik cloud IoT platform is developed by Samsung to enable devices to connect to cloud services. It has a set of services that continuously connect devices to the cloud and start gathering data. It stores the incoming data from connected devices and combines this information. This platform contains a set of connectors that connect to third-party services.

How IoT platform help

- IoT Platform connects sensors and devices.
- IoT platform handles different software communication protocol and hardware.
- IoT platform provides security and authentication for sensors and users.

- It collects, visualizes, and analyzes the data gathered by the sensor and device.

NODE MCU

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS.

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266.^[10] The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects). NodeMCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub.

ESP8266 Arduino Core

As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate toolchains to allow Arduino C/C++ to be compiled for these new processors. They did this with the introduction of the Board Manager and the SAM Core. A "core" is the collection of software components required by the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file for the target MCU's machine language. This has become a leading software development platform for the various ESP8266-based modules and development boards, including NodeMCUs.

Introduction

NodeMCU is an open source LUA based firmware developed for ESP8266 wifi chip. By exploring functionality with ESP8266 chip, NodeMCU

firmware comes with ESP8266 Development board/kit i.e. NodeMCU Development board.

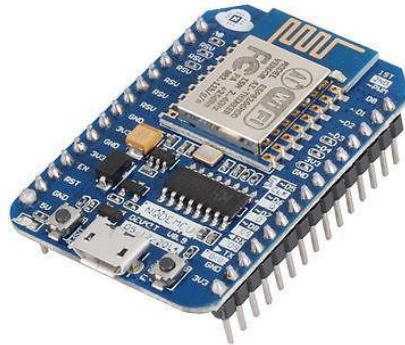


Fig. 4.17: NodeMCU

NodeMCU Development Board/kit v0.9 (Version1)

Since NodeMCU is open source platform, their hardware design is open for edit/modify/build.

NodeMCU Dev Kit/board consist of ESP8266 wifi enabled chip. The **ESP8266** is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer ESP8266 WiFi Module.

There is Version2 (V2) available for NodeMCU Dev Kit i.e. **NodeMCU Development Board v1.0 (Version2)**, which usually comes in black colored PCB.



Fig. 4.18: NodeMCU Development Board/kit v0.9

NodeMCU Development Board/kit v1.0 (Version2)

For more information about NodeMCU Boards available in market refer NodeMCU Development Boards

NodeMCU Dev Kit has **Arduino like** Analog (i.e. A0) and Digital (D0-D8) pins on its board. It supports serial communication protocols i.e. UART, SPI, I2C etc.

Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc.

How to start with NodeMCU?

NodeMCU Development board is featured with wifi capability, analog pin, digital pins and serial communication protocols.

To get start with using NodeMCU for IoT applications first we need to know about how to write/download NodeMCU firmware in NodeMCU Development Boards. And before that where this NodeMCU firmware will get as per our requirement.

To know more about how to build custom NodeMCU firmware online and download it refer Getting started with NodeMCU

How to write codes for NodeMCU?

After setting up ESP8266 with Node-MCU firmware, let's see the IDE (Integrated Development Environment) required for development of NodeMCU.

NodeMCU with ESPlorer IDE

Lua scripts are generally used to code the NodeMCU. Lua is an open source, lightweight, embeddable scripting language built on top of C programming language.

For more information about how to write Lua script for NodeMCU refer Getting started with NodeMCU using ESPlorerIDE

NodeMCU with Arduino IDE

Here is another way of developing NodeMCU with a well-known IDE i.e. Arduino IDE. We can also develop applications on NodeMCU using Arduino development environment. This makes easy for Arduino developers than learning new language and IDE for NodeMCU.

For more information about how to write Arduino sketch for NodeMCU refer Getting started with NodeMCU using ArduinoIDE

ESP32-CAM

The ESP32-CAM is a very small camera module with the ESP32-S chip that costs approximately \$10. Besides the OV2640 camera, and several GPIOs to connect peripherals, it also features a microSD card slot that can be useful to store images taken with the camera or to store files to serve to clients.

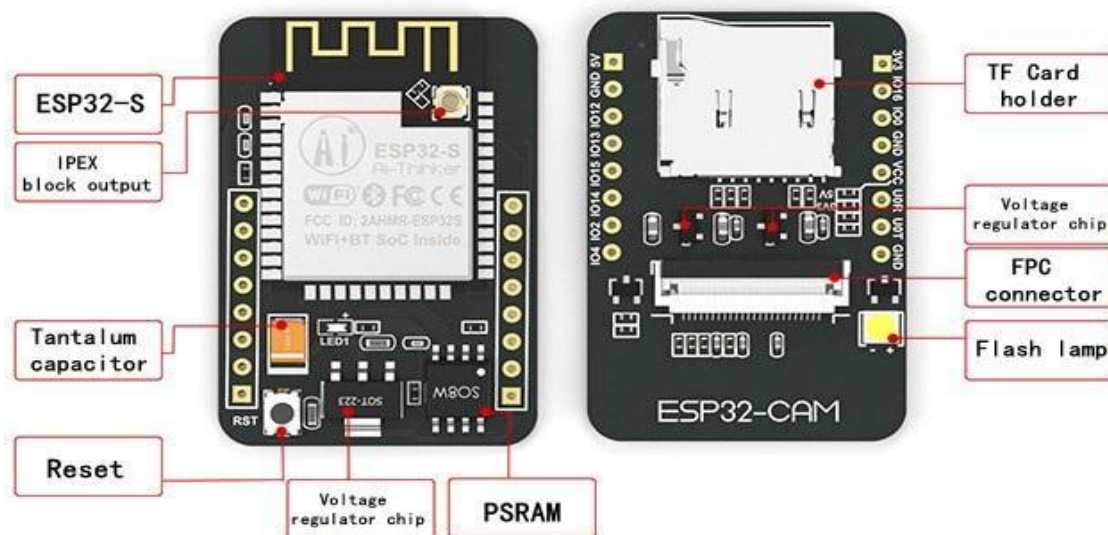


Fig. 4.19: Image source – Seed Studio

The ESP32-CAM doesn't come with a USB connector, so you need an FTDI programmer to upload code through the **U0R** and **U0T** pins (serial pins).

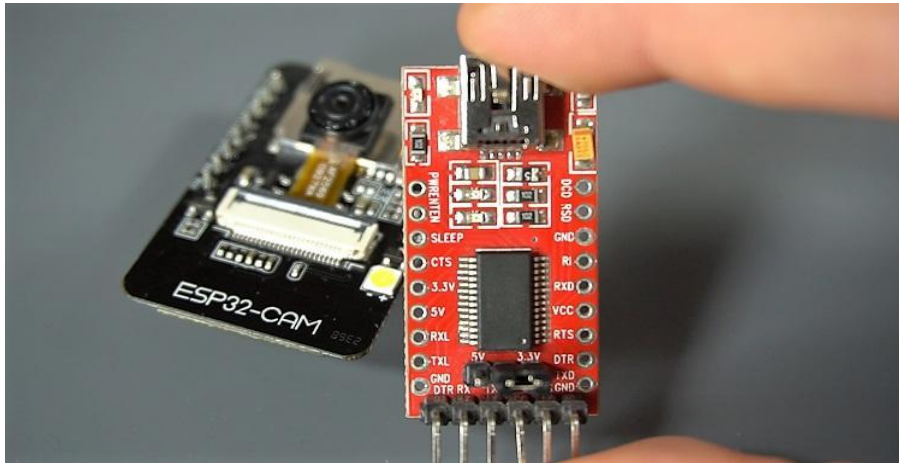


Fig. 4.20: ESP32-CAM

Features

Here is a list with the ESP32-CAM features:

- The smallest 802.11b/g/n Wi-Fi BT SoC module
- Low power 32-bit CPU, can also serve the application processor
- Up to 160MHz clock speed, summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras, built-in flash lamp
- Support image WiFi upload
- Support TF card
- Supports multiple sleep modes
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)

ESP32-CAM Pinout

The following figure shows the ESP32-CAM pinout (AI-Thinker module).

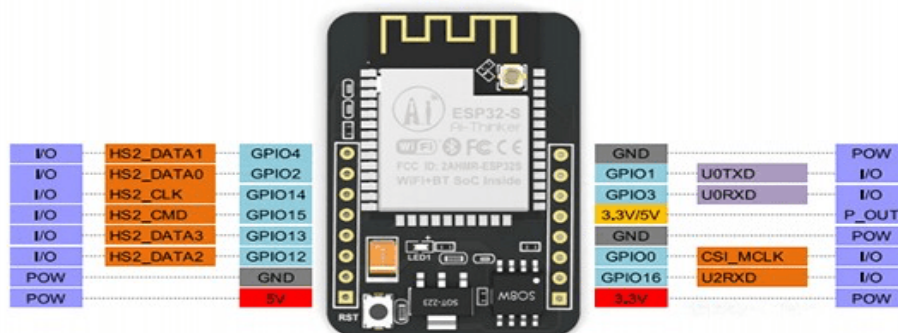


Fig. 4.21: Image source – Seed Studio

There are three **GND** pins and two pins for power: either **3.3V** or **5V**. **GPIO 1** and **GPIO 3** are the serial pins. You need these pins to upload code to your board. Additionally, **GPIO 0** also plays an important role, since it determines whether the ESP32 is in flashing mode or not. When **GPIO 0** is connected to **GND**, the ESP32 is in flashing mode.

The following pins are internally connected to the microSD card reader:

- GPIO 14: CLK
- GPIO 15: CMD
- GPIO 2: Data 0
- GPIO 4: Data 1 (also connected to the on-board LED)
- GPIO 12: Data 2
- GPIO 13: Data 3

Video Streaming Server

Follow the next steps to build a video streaming web server with the ESP32-CAM that you can access on your local network.

Important: Make sure you have your Arduino IDE updated as well as the latest version of the ESP32 add-on.

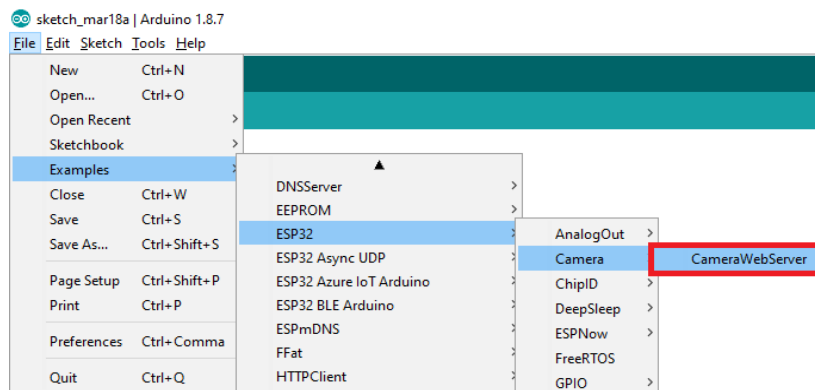
1. *Install the ESP32 add-on*

In this example, we use Arduino IDE to program the ESP32-CAM board. So, you need to have Arduino IDE installed as well as the ESP32 add-on. Follow one of the next tutorials to install the ESP32 add-on, if you haven't already:

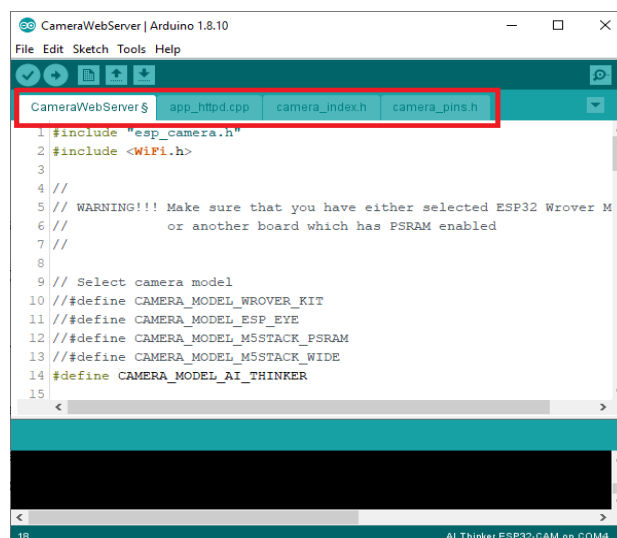
- Installing the ESP32 Board in Arduino IDE (Windows instructions)
- Installing the ESP32 Board in Arduino IDE (Mac and Linux instructions)

2. *Camera Webserver Example Code*

In your Arduino IDE, go to **File > Examples > ESP32 > Camera** and open the **Camera Webserver** example.



The following code should load



Before uploading the code, you need to insert your network credentials in the following variables:

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";  
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

Then, make sure you select the right camera module. In this case, we're using the AI-THINKER Model.

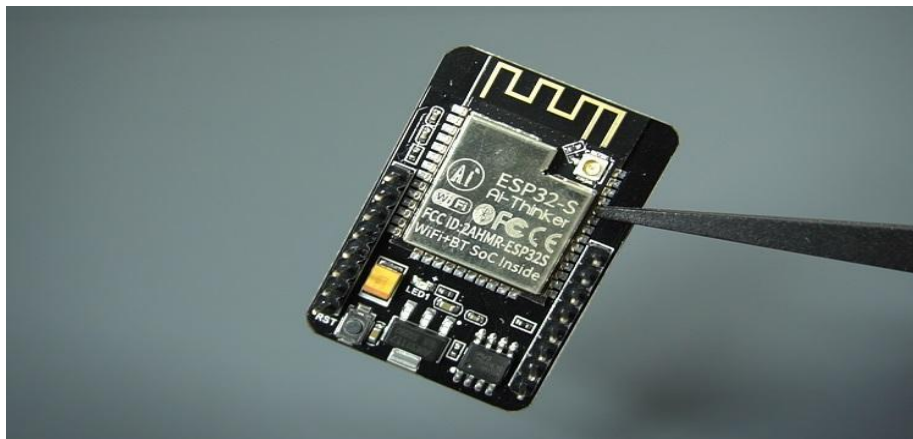


Fig. 4.22: AI-THINKER Model

So, comment all the other models and uncomment this one:

```
// Select camera model  
  
//#define CAMERA_MODEL_WROVER_KIT  
  
//#define CAMERA_MODEL_ESP_EYE  
  
//#define CAMERA_MODEL_M5STACK_PSRAM  
  
//#define CAMERA_MODEL_M5STACK_WIDE  
  
#define CAMERA_MODEL_AI_THINKER
```

If none of these correspond to the camera you're using, you need to add the pin assignment for your specific board in the camera_pins.h tab.

Now, the code is ready to be uploaded to your ESP32.

4) When you start to see these dots on the debugging window as shown below, press the ESP32-CAM on-board RST button.

```
esptool.py v2.6-beta1
Serial port COM10
Connecting.....
```

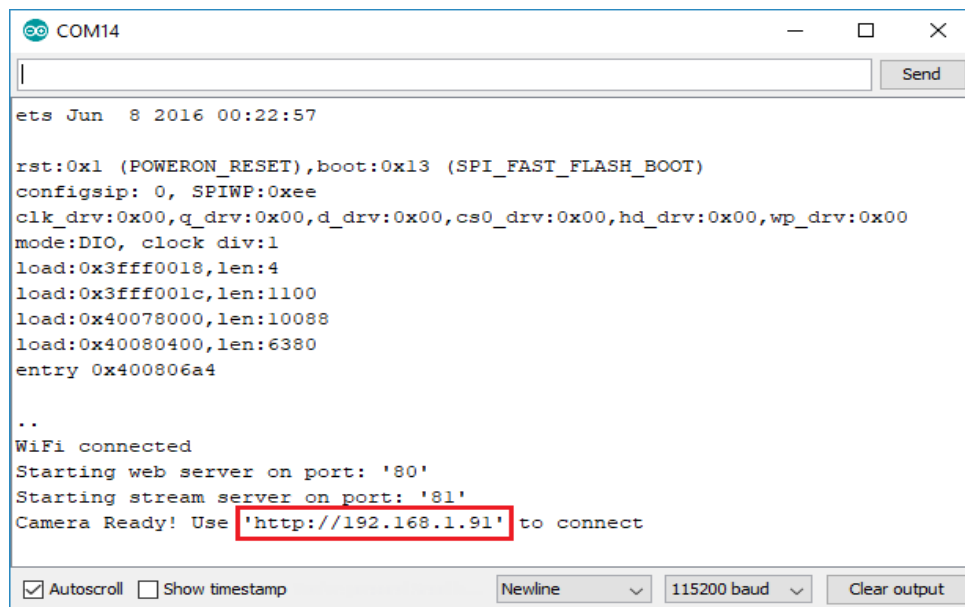
After a few seconds, the code should be successfully uploaded to your board.

Getting the IP address

After uploading the code, disconnect GPIO 0 from GND.

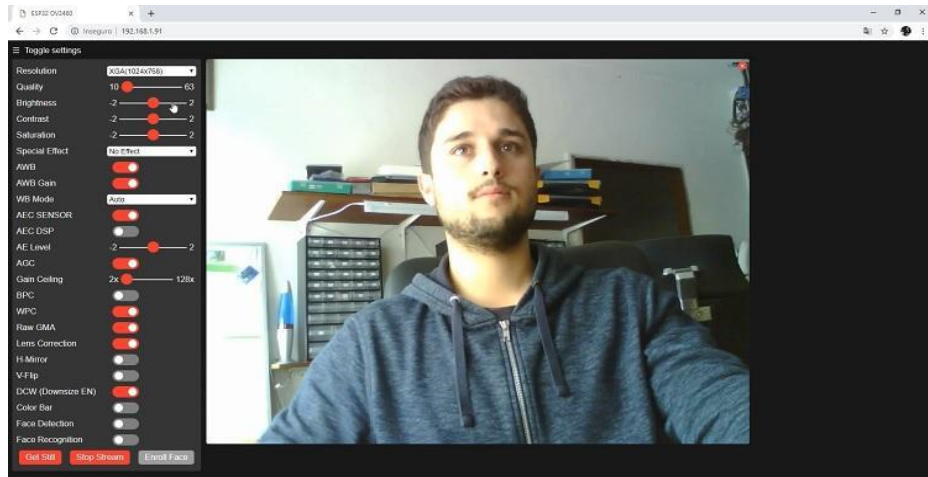
Open the Serial Monitor at a baud rate of 115200. Press the ESP32-CAM on-board Reset button.

The ESP32 IP address should be printed in the Serial Monitor.



Accessing the Video Streaming Server

Now, you can access your camera streaming server on your local network. Open a browser and type the ESP32-CAM IP address. Press the **Start Streaming** button to start video streaming.



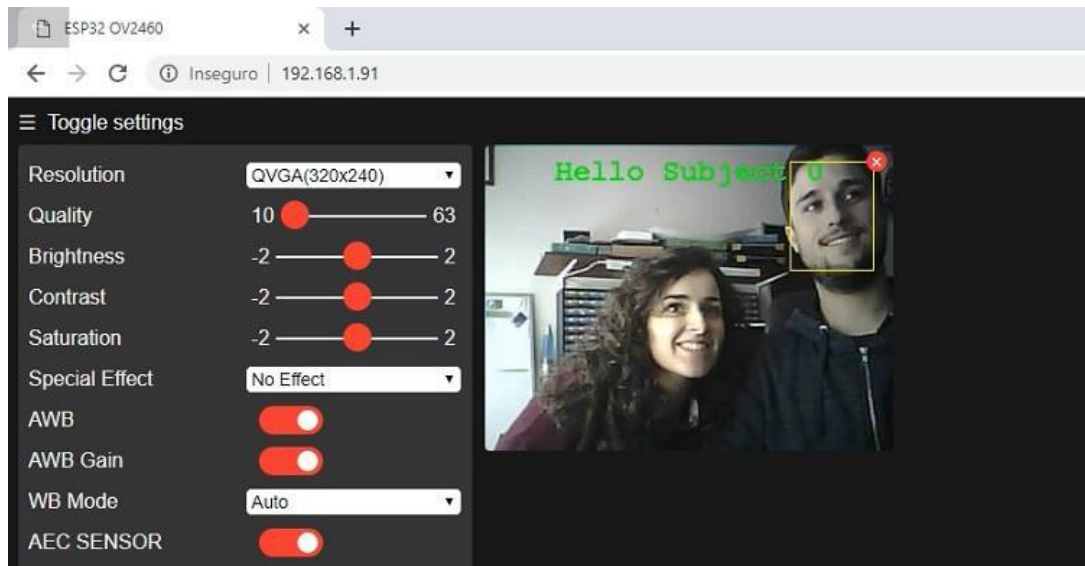
You also have the option to take photos by clicking the **Get Still** button. Unfortunately, this example doesn't save the photos, but you can modify it to use the on board microSD Card to store the captured photos.

There are also several camera settings that you can play with to adjust the image settings.

Finally, you can do face recognition and detection.



First, you need to enroll a new face. It will make several attempts to save the face. After enrolling a new user, it should detect the face later on (subject 0).



And that's it. Now you have your video streaming web server up and running with face detection and recognition with the example from the library.

Troubleshooting

If you're getting any of the following errors, read our **ESP32-CAM Troubleshooting Guide: Most Common Problems Fixed**

- Failed to connect to ESP32: Timed out waiting for packet header
- Camera init failed with error 0x20001 or similar
- Brownout detector or Guru meditation error
- Sketch too big error – Wrong partition scheme selected
- Board at COMX is not available – COM Port Not Selected
- Psram error: GPIO isr service is not installed
- Weak Wi-Fi Signal
- No IP Address in Arduino IDE Serial Monitor
- Can't open web server
- The image lags/shows lots of latency

Humidity sensor

Measurement and control of temperature and relative humidity finds applications in numerous areas. These days devices are available which have both temperature and humidity sensors with signal conditioning, ADC, calibration and communication interface all built inside them. The use of such smart sensors greatly simplify the design and reduces the overall cost. We discussed in past about Humidity and temperature measurements with Sensirion's SHT1x/SHT7x sensors. The DHT11 sensor uses a proprietary 1-wire protocol which we will be exploring here and implementing with the PIC16F628A microcontroller that will receive the temperature and humidity values from the sensor and display them on a 16×2 character LCD.

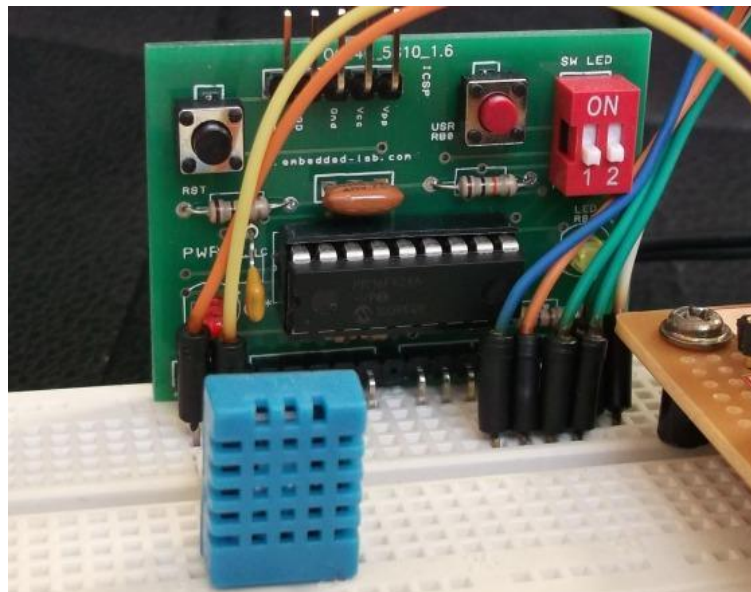


Fig. 4.24: Interfacing DHT11 sensor with PIC16F628A

About DHT11 sensor

The DHT11 sensor comes in a single row 4-pin package and operates from 3.5 to 5.5V power supply. It can measure temperature from 0-50 °C with an accuracy of $\pm 2^{\circ}\text{C}$ and relative humidity ranging from 20-95% with an accuracy of $\pm 5\%$. The sensor provides fully calibrated digital outputs for the two measurements. It has got its own proprietary 1-wire protocol, and therefore, the communication between the sensor and a microcontroller is not possible through a direct interface

with any of its peripherals. The protocol must be implemented in the firmware of the MCU with precise timing required by the sensor.

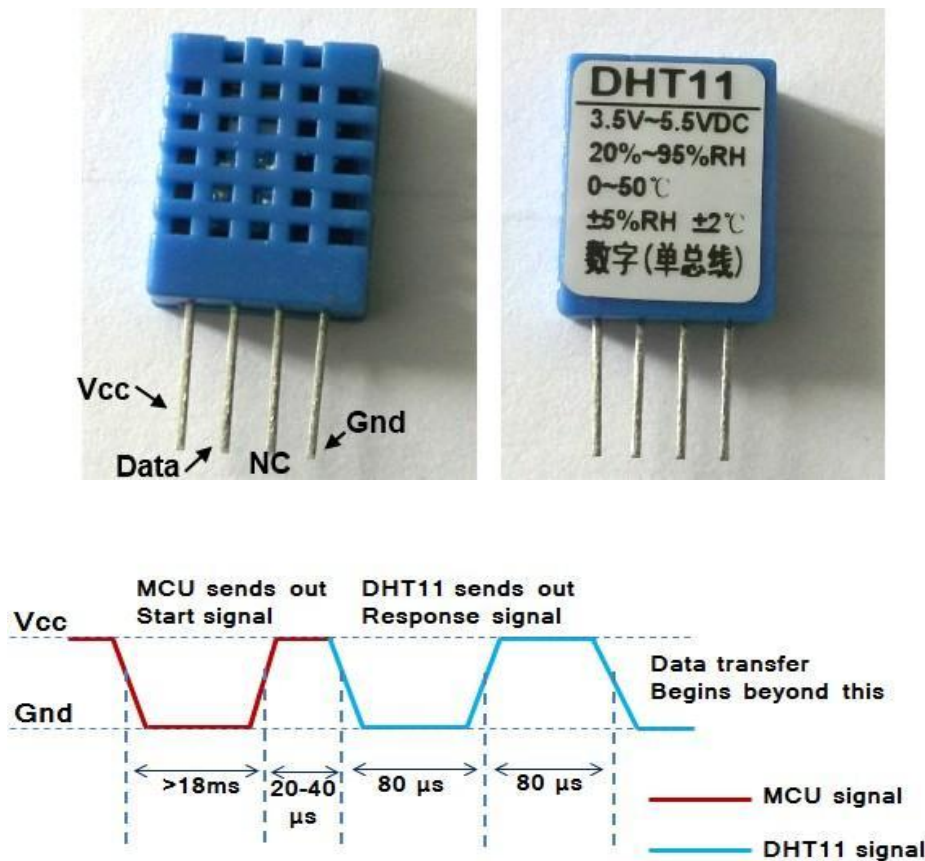


Fig. 4.25: "Start" and "Response" signals

The 40-bit data from the sensor has the following structure.

Data (40-bit) = Integer Byte of RH + Decimal Byte of RH + Integer Byte of Temp. + Decimal Byte of Temp. + Checksum Byte

For DHT11 sensor, the decimal bytes of temperature and humidity measurements are always zero. Therefore, the first and third bytes of received data actually give the numeric values of the measured relative humidity (%) and temperature (°C). The last byte is the checksum byte which is used to make sure that the data transfer has happened without any error. If all the five bytes are transferred successfully then the checksum byte must be equal to the last 8 bits of the sum of the first four bytes, i.e.,

Checksum = Last 8 bits of (Integer Byte of RH + Decimal Byte of RH + Integer Byte of Temp. + Decimal Byte of Temp.)

Now let's talk about the most important thing, which is signalling for transmitting "0" and "1". In order to send a bit of data, the sensor first pulls the line low for 50 µs. Then it raises the line to high for 26-28 µs if it has to send "0", or for 70 µs if the bit to be transmitted is "1". So it is the width of the positive pulse that carries information about 1 and 0.

Timing difference for transmitting "1s" and "0s"

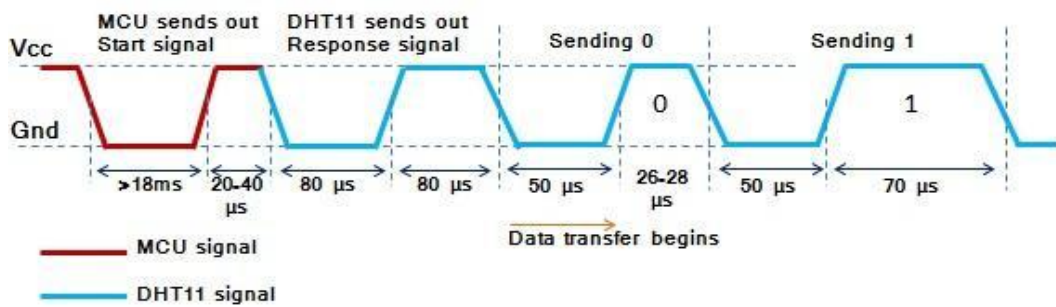


Fig. 4.26: Start, Response and Data signals in sequence

At the end of the last transmitted bit, the sensor pulls the data line low for 50 µs and then releases it. The DHT11 sensor requires an external pull-up resistor to be connected between its Vcc and the data line so that under idle condition, the data line is always pulled high. After finishing the data transmission and releasing the data line, the DHT11 sensor goes to the low-power consumption mode until a new "Start" signal arrives from the MCU.

Circuit diagram

Here is the circuit diagram showing the DHT11 sensor and a HD44780-based character LCD interfaced to the PIC16F628A microcontroller. The microcontroller runs at 4.0 MHz clock using an external resonator connected between OSC1 (16) and OSC2 (15) pins. The use of 4.0 MHz clock makes the timing calculation easier as 1 machine cycle becomes 1 µs. The timing information will be

used to calculate the width of the received data pulse from the sensor so that we could identify if it is carrying a 1 or 0.

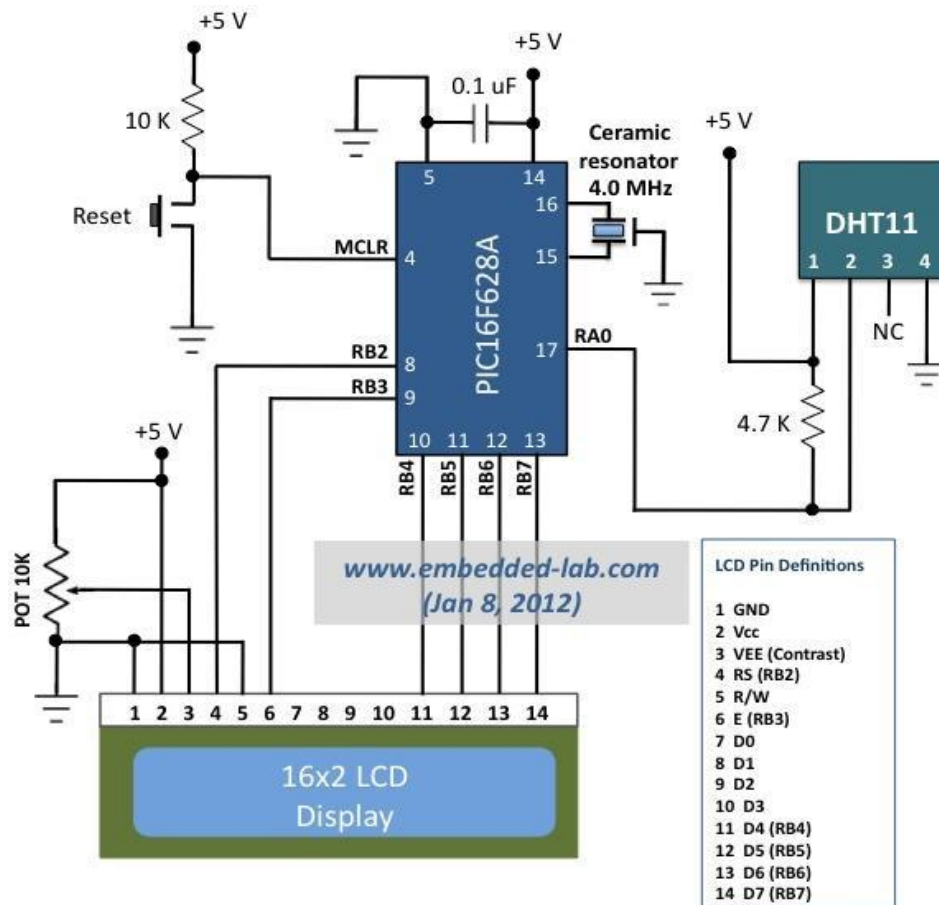


Fig. 4.27: Circuit connections for PIC16F628A and DHT11 sensor

The following pictures show the circuit setup on a breadboard. Don't get confused with the four LEDs and tact switches shown on the perforated board. They have nothing to do with this project. They are there because I am using my **DIY Experimenter's I/O board** for the LCD part of this project. Similarly, I am using my **18-pin PIC16F board** for easy prototyping with the PIC16F628A microcontroller.

Soil Moisture Sensor

Soil moisture sensors measure the volumetric water content in soil.[1] Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighting of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content. The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Reflected microwave radiation is affected by the soil moisture and is used for remote sensing in hydrology and agriculture. Portable probe instruments can be used by farmers or gardeners.

Soil moisture sensors typically refer to sensors that estimate volumetric water content. Another class of sensors measure another property of moisture in soils called water potential; these sensors are usually referred to as soil water potential sensors and include tensiometers and gypsum blocks.

Research

Soil moisture sensors are used in numerous research applications, e.g. in agricultural science and horticulture including irrigation planning, climate research, or environmental science including solute transport studies and as auxiliary sensors for soil respiration measurements.

Simple sensors for gardeners

Relatively cheap and simple devices that do not require a power source are available for checking whether plants have sufficient moisture to thrive. After inserting a probe into the soil for approximately 60 seconds, a meter indicates if the soil is too dry, moist or wet for plant.

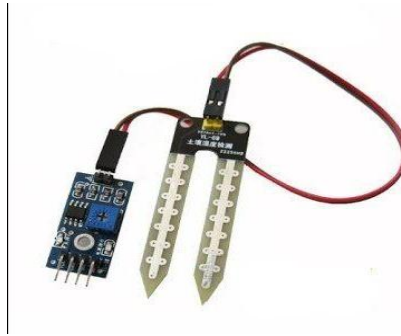


Fig. 4.29: Simple sensors for gardeners

Gas sensor module

In current technology scenario, monitoring of gases produced is very important. From home appliances such as air conditioners to electric chimneys and safety systems at industries monitoring of gases is very crucial. **Gas sensors** are very important part of such systems. Small like a nose, gas sensors spontaneously react to the gas present, thus keeping the system updated about any alterations that occur in the concentration of molecules at gaseous state.

Gas sensors are available in wide specifications depending on the sensitivity levels, type of gas to be sensed, physical dimensions and numerous other factors. This Insight covers a **methane gas sensor** that can sense gases such as ammonia which might get produced from methane. When a gas interacts with this sensor, it is first ionized into its constituents and is then adsorbed by the sensing element. This adsorption creates a potential difference on the element which is conveyed to the processor unit through output pins in form of current. What is this sensing element? Is it kept in some chamber or is kept exposed? How does it get current and how it is taken out? Let's find out in this Insight!!!



Fig. 4.30: Gas sensor module

Image shows externals of a standard gas sensor module: a steel mesh, copper clamping ring and connecting leads. The top part is a stainless steel mesh which takes care of the following:

Filtering out the suspended particles so that only gaseous elements are able to pass to insides of the sensor.

- ✓ Protecting the insides of the sensor.
- ✓ Exhibits an anti-explosion network that keeps the sensor module intact at high temperatures and gas pressures.

In order to manage above listed functions efficiently, the steel mesh is made into two layers. The mesh is bound to rest of the body via a copper plated clamping ring.

The connecting leads of the sensor are thick so that sensor can be connected firmly to the circuit and sufficient amount of heat gets conducted to the inside part. They are casted from copper and have tin plating over them. Four of the six leads (A, B, C, D) are for signal fetching while two (1, 2) are used to provide sufficient heat to the sensing element.

The pins are placed on a Bakelite base which is a good insulator and provides firm gripping to the connecting leads of the sensor.

The top of the gas sensor is removed off to see the internals parts of the sensor: sensing element and connection wiring. The hexapod structure is constituted by the sensing element and six connecting legs that extend beyond the Bakelite base.

Using a ceramic substrate increases the heating efficiency and tin oxide, being sensitive towards adsorbing desired gas' components (in this case methane and its products) suffices as sensing coating.

Sound Sensor Basics

Pin Configuration, Working, Applications and Interfacing

In this fully automated world, we want to operate the things through the sound for our easyness. To make this automation, we need **Sound sensor**. In this article, we are going to learn about the basics of sound sensor. Also, here we will learn about the interfacing sound sensor with arduino.

What is Sound Sensor?

Basically, Sound is a waveform of energy which is produced by the form of mechanical vibration. The type of sound determines its frequency. For example, a bass drum has a low-frequency sound and a cymbal has a higher frequency sound. The sound sensor is a simple device which can detect the sound. The sound sensors are very simple to use.

Pin Configuration and Components

The sound sensor consist of Microphone as a transducer, potentiometer to adjust the intensity, LM386 low power audio amplifier, LED and other passive components like resistors and capacitors.

You can set a threshold value using a potentiometer so that when the amplitude of the sound exceeds the threshold value, the module will output LOW otherwise HIGH.

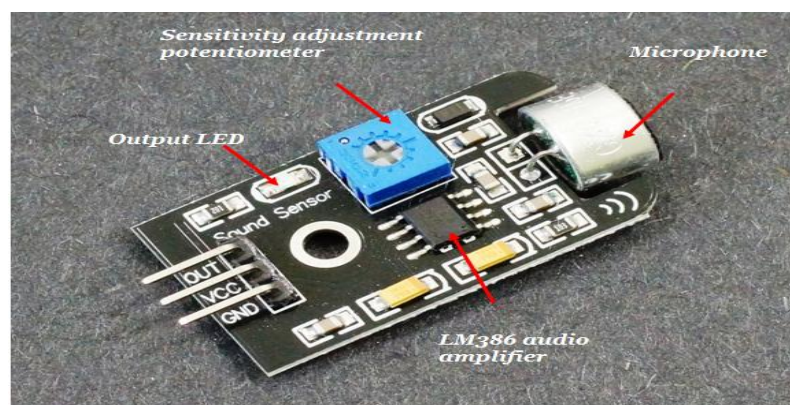


Fig.: 4.31: Sound sensor consist of Microphone

This sensor includes 3 pins and they are,

- Pin1 (VCC): 3.3V DC to 5V DC
- Pin2 (GND): This is a ground pin
- Pin3 (OUT): This is an output pin. It provides high signal when there is no sound and goes LOW when sound is detected. You can connect it to any digital pin on an Arduino or directly to a 5V relay or similar device.

Working Principle of Sound Sensor

It works similar to our ears. Our Ears have a diaphragm which converts the detected vibration and converts it into the signal. Similarly, the sound sensors convert the vibration into audio signal (voltage and current proportional) with the help of a microphone. This microphone has an inbuilt diaphragm, made up of magnets which are coiled by metal wire. Whenever sound waves hit the diaphragm, magnets vibrate and at the same time coil induces the current.

Interfacing of Sound Sensor with Arduino Uno

Let's hook the sound sensor with Arduino Uno. Connections are small and simple. Here we connect Vcc to 5V, Gnd to Gnd of Arduino Uno and connect the OUT pin to the D8 pin of Arduino. That's it! For output indication, we connect the 100k resistor and one LED to D12.

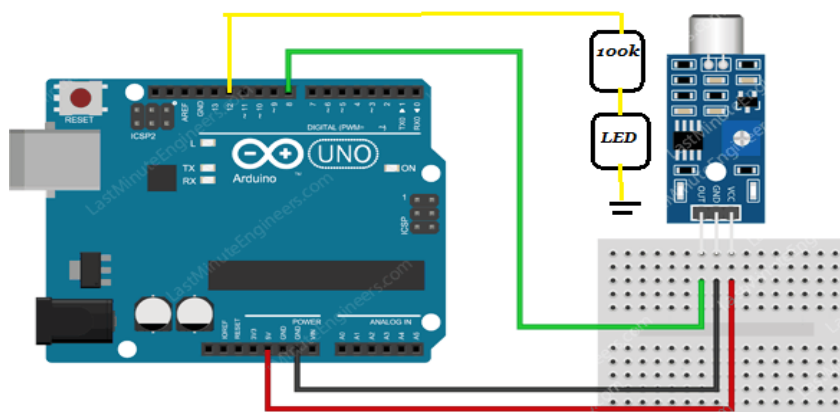


Fig. 4.32: Arduino Uno

Download the Arduino IDE from the below link and use the below code.

Arduino IDE 1.8.13

```
const int ledPin = 12;
const int soundPin = 8;

int soundVal = 0;

void setup ()
{
  pinMode (ledPin, OUTPUT);
  pinMode (soundPin, INPUT);
  Serial.begin (9600);
}

void loop ()
{
  soundVal = digitalRead(soundPin);
  if (soundVal == HIGH)
  {
    digitalWrite(ledPin, HIGH);
    Serial.println("Clap detected");
    delay(1000);
  }
  else
  {
    digitalWrite(ledPin,LOW);
  }
}
```

The below GIF shows that the LED is ON, as the clap is detected.

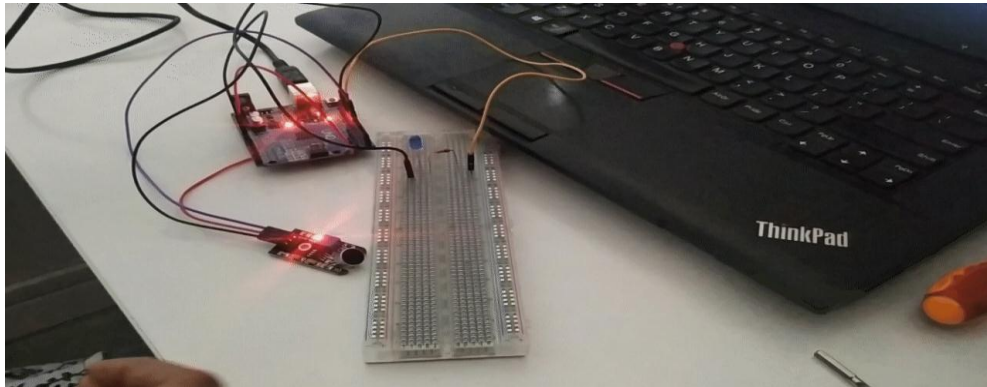
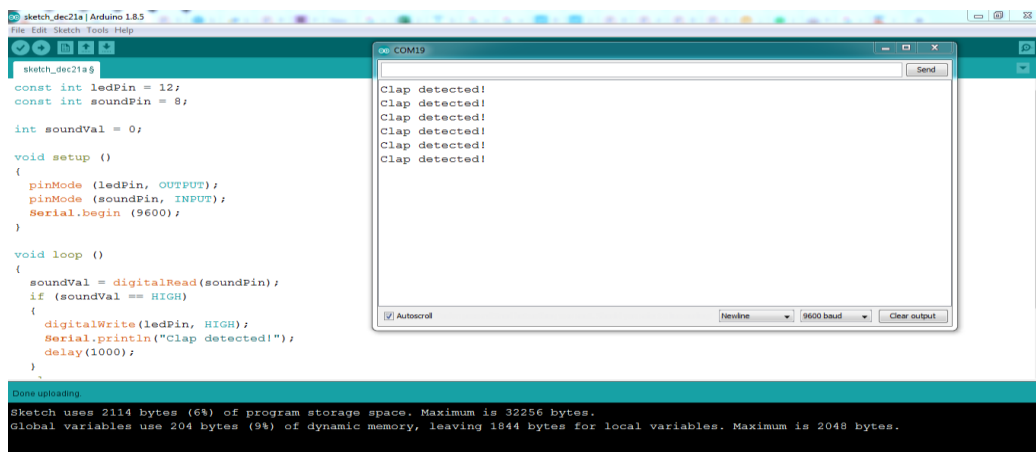


Fig. 4.33: GIF shows that the LED is ON, as the clap

Simultaneously, you can see the output on the serial window of Arduino IDE.



Applications of Sound Sensor

- Security system for Office or Home
- Spy Circuit
- Home Automation
- Robotics
- Smart Phones
- Ambient sound recognition
- Audio amplifier
- Sound level recognition

This blog discusses the complete overview of the sound sensor. Hope this article helps you to understand the working principle, Interfacing and Applications of sound sensor.

Servo Motors

A **servomechanism**, or **servo**, is an automatic device that uses error-sensing negative feedback to correct the performance of a mechanism. The term correctly applies only to systems where the feedback or error-correction signals help control mechanical position or other parameters. For example, the car's cruise control uses closed loop feedback, which classifies it as a servomechanism.

RC servos are hobbyist remote control devices servos typically employed in radio-controlled models, where they are used to provide actuation for various mechanical systems. Due to their affordability, reliability, and simplicity of control by microprocessors, RC servos are often used in small-scale robotics applications.

RC servos are composed of an electric motor mechanically linked to a potentiometer. A standard RC receiver sends Pulse-width modulation (PWM) signals to the servo. The electronics inside the servo translate the width of the pulse into a position. When the servo is commanded to rotate, the motor is powered until the potentiometer reaches the value corresponding to the commanded position.

RC servos use a three-pin 0.1" spacing jack (female) which mates to standard 0.025" square pins (which should be gold-plated, incidentally). The most common order is Signal, +voltage, ground. The standard voltage is 6VDC, however 4.8V and 12V has also been seen for a few servos. The control signal is a digital PWM signal with a 50Hz frame rate. Within each 20ms timeframe, an active-high digital pulse controls the position. The pulse nominally ranges from 1.0ms to 2.0ms with 1.5ms always being center of range. Pulse widths outside this range can be used for "over travel" -moving the servo beyond its normal range. This PWM signal sometimes (incorrectly) called Pulse Position Modulation (PPM).

The servo is controlled by three wires: ground, power, and control. The servo will move based on the pulses sent over the control wire, which set the angle of the actuator arm.

The servo expects a pulse every 20 ms in order to gain correct information about the angle. The width of the servo pulse dictates the range of the servo's angular motion.

RC servos are usually powered by the receiver which in turn is powered by battery packs or an Electronic speed controller (ESC) with an integrated or a separate Battery eliminator circuit (BEC). Common battery packs are either Ni-Cd, NiMH or lithium-ion polymer battery (Li-Po) type. Voltage ratings vary, but most receivers are operated at 5 V or 6 V.

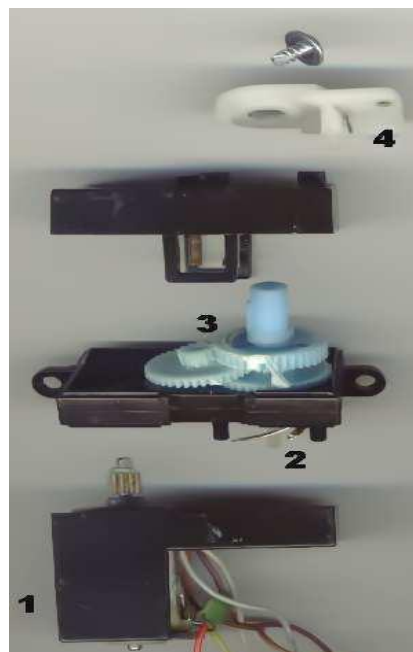


Fig. 4.34: Small R/C servo mechanism

- | | |
|------------------------------------|-------------------|
| 1. Electric motor | 3. Reduction gear |
| 2. Position feedback potentiometer | 4. Actuator arm |

4.2 SOFTWARE REQUIRED

Arduino Software (IDE)

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Writing Sketches

Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.



Verify

Checks your code for errors compiling it.



Upload

Compiles your code and uploads it to the configured board. See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"



New

Creates a new sketch.



Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the **File | Sketchbook** menu instead.



Save

Saves your sketch.



Serial Monitor

Opens the serial monitor.

Additional commands are found within the five menus: **File**, **Edit**, **Sketch**, **Tools**, **Help**. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

File

- *New*
Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.
- *Open*
Allows to load a sketch file browsing through the computer drives and folders.
- *Open Recent*
Provides a short list of the most recent sketches, ready to be opened.
- *Sketchbook*
Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.
- *Examples*
Any example provided by the Arduino Software (IDE) or library shows up in this

menu item. All the examples are structured in a tree that allows easy access by topic or library.

- *Close*
Closes the instance of the Arduino Software from which it is clicked.
- *Save*
Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.
- *Save as...*
Allows to save the current sketch with a different name.
- *Page Setup*
It shows the Page Setup window for printing.
- *Print*
Sends the current sketch to the printer according to the settings defined in Page Setup.
- *Preferences*
Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.
- *Quit*
Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

Edit

- *Undo/Redo*
Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.
- *Cut*
Removes the selected text from the editor and places it into the clipboard.
- *Copy*
Duplicates the selected text in the editor and places it into the clipboard.

- *Copy for Forum*
Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- *Copy as HTML*
Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- *Paste*
Puts the contents of the clipboard at the cursor position, in the editor.
- *Select All*
Selects and highlights the whole content of the editor.
- *Comment/Uncomment*
Puts or removes the // comment marker at the beginning of each selected line.
- *Increase/Decrease Indent*
Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.
- *Find*
Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.
- *Find Next*
Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.
- *Find Previous*
Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

Sketch

- *Verify/Compile*
Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

- *Upload*
Compiles and loads the binary file onto the configured board through the configured Port.
- *Upload Using Programmer*
This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a *Tools -> Burn Bootloader* command must be executed.
- *Export Compiled Binary*
Saves a .hex file that may be kept as archive or sent to the board using other tools.
- *Show Sketch Folder*
Opens the current sketch folder.
- *Include Library*
Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.
- *Add File...*
Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side of the toolbar.

Tools

- *Auto Format*
This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
- *Archive Sketch*
Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

- *Fix Encoding & Reload*

Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

- *Serial Monitor*

Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

- *Board*

Select the board that you're using. See below for descriptions of the various boards.

- *Port*

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

- *Programmer*

For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

- *Burn Bootloader*

The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the **Boards** menu before burning the bootloader on the target board. This command also set the right fuses.

Help

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The

documents are a local copy of the online ones and may link back to our online website.

- *Find in Reference*

This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

Sketchbook

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the **File > Sketchbook** menu or from the **Open** button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the **Preferences** dialog.

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

Uploading

Before uploading your sketch, you need to select the correct items from the **Tools > Board** and **Tools > Port** menus. The boards are described below. On the Mac, the serial port is probably something like **/dev/tty.usbmodem241** (for an Uno or Mega2560 or Leonardo) or **/dev/tty.usbserial-1B1** (for a Duemilanove or earlier USB board), or **/dev/tty.USA19QW1b1P1.1** (for a serial board connected with

a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be **/dev/ttyACMx**, **/dev/ttyUSBx** or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the **Upload** item from the **Sketch** menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the **Sketch > Import Library** menu. This will insert one or more **#include** statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its **#include** statements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

To write your own library, see this tutorial.

Serial Monitor

This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the

rate passed to **Serial.begin** in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

Preferences

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog

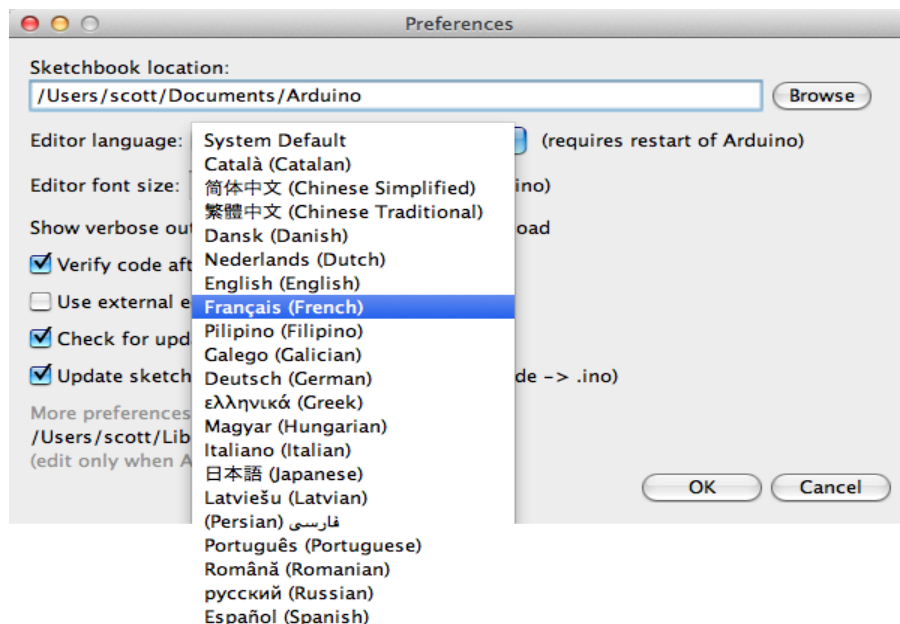


Fig. 4.35: Preferences

Since version 1.0.1, the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: on Windows and possibly Linux, this is determined by the

locale setting which controls currency and date formats, not by the language the operating system is displayed in.)

You can return the software to its default setting of selecting its language based on your operating system by selecting **System Default** from the **Editor Language** drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

Boards

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets and the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the bootloader. You can find a comparison table between the various boards [here](#).

- *Arduino* *Yùn*
An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino/Genuino Uno*
An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Diecimila or Duemilanove w/ ATmega168*
An ATmega168 running at 16 MHz with auto-reset.
- *Arduino Nano w/ ATmega328P*
An ATmega328P running at 16 MHz with auto-reset. Has eight analog inputs.
- *Arduino/Genuino Mega 2560*
An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.

- *Arduino Mega*
An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino Mega ADK*
An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino Leonardo*
An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino/Genuino Micro*
An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino Esplora*
An ATmega32u4 running at 16 MHz with auto-reset.
- *Arduino Mini w/ ATmega328P*
An ATmega328P running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Ethernet*
Equivalent to Arduino UNO with an Ethernet shield: An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Fio*
An ATmega328P running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328P, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino BT w/ ATmega328P*
ATmega328P running at 16 MHz. The bootloader burned (4 KB) includes codes to initialize the on-board bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM.
- *LilyPad Arduino USB*
An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.

- *LilyPad Arduino*

An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

- *Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328P*

An ATmega328P running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328P; 6 Analog In, 14 Digital I/O and 6 PWM.

- *Arduino NG or older w/ ATmega168*

An ATmega168 running at 16 MHz *without* auto-reset. Compilation and upload is equivalent to Arduino Diecimila or Duemilanove w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.

- *Arduino Robot Control*

An ATmega328P running at 16 MHz with auto-reset.

- *Arduino Robot Motor*

An ATmega328P running at 16 MHz with auto-reset.

- *Arduino Gemma*

An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

For instructions on installing support for other boards, see third-party hardware above.

Last revision 2015/09/07 by SM

The text of the Arduino getting started guide is licensed under a Creative Commons Attribution-ShareAlike 3.0 License. Code samples in the guide are released into the public domain.

4.3 UBIDOTS EXPLORER

Introduction

Welcome to the Ubidots REST API hardware docs. Here you would reference how to send or retrieve data from your hardware devices using our supported communication protocols: **HTTP**, **MQTT** and **TCP/UDP**.

The purpose of this section is to help you understand what happens in the backstage when communicating with Ubidots, so you can replicate this in your firmware. For this reason, we avoid the use of examples using libraries. However, if you'd like to use one of our libraries (Python, C, and more), check out our compatible devices and API clients sections.

How Ubidots Works?

Every time a device updates a sensor value in a variable, a data-point or "**dot**" is created. Ubidots stores dots that come from your devices inside variables, and these stored dots have corresponding **timestamps**:

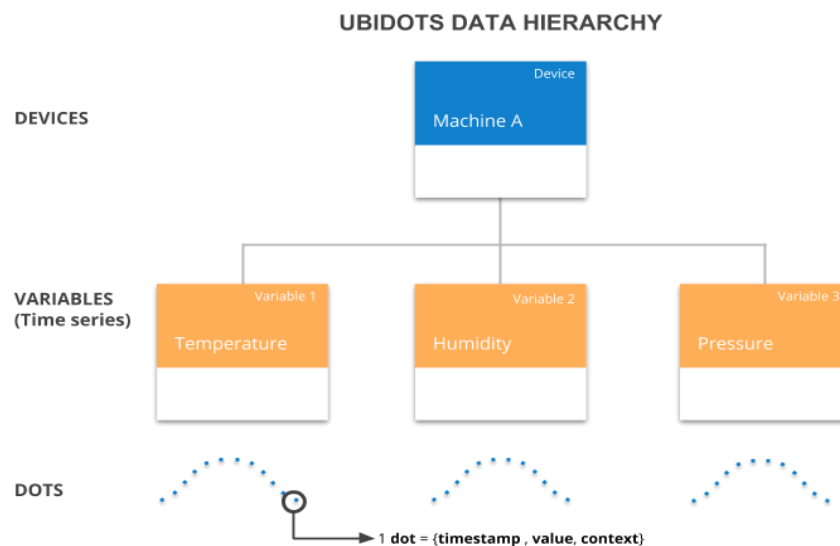


Fig. 4.36: Ubidots Data Hierarchy

Each dot contains these items

Item	Description	Mandatory
value	A numerical value. Ubidots accepts up to 16 floating-point length numbers.	YES
timestamp	Unix Epoch time, in milliseconds. If not specified, then our servers will assign one upon reception.	NO
context	An arbitrary collection of key-value pairs. Mostly used to store the latitude and longitude coordinates of GPS devices.	NO

Values

A numerical value. Ubidots accepts up to 16 floating-point length numbers.

```
{"value" : 34.87654974}
```

Timestamps

A **timestamp**, as best described here, is a way to track time as a running total of seconds. This count starts at the Unix Epoch on January 1st, 1970 at UTC. Therefore, the unix time stamp is merely the number of seconds between a particular date and the Unix Epoch. Please keep in mind that when you send data to Ubidots, you must set the timestamp in milliseconds; also, if you retrieve a dot's timestamp, it will be in milliseconds.

```
"timestamp" : 1537453824000
```

The above timestamp corresponds to Thursday, September 20, 2018 2:30:24 PM.

PRO-TIP: A useful tool to convert between Unix timestamps and human-readable dates is Epoch Converter.

Context

Numerical values are not the only data type supported; you can also store *string* or *char* data types inside what we call **context**. The **context** is a key-

value object that allows you to store not only numerical but also string values. An example use of the **context** could be:

```
"context" : {"status" : "on", "weather" : "sunny"}
```

A **context** is commonly used to store the latitude and longitude coordinates of your device for GPS/tracking application use cases. All Ubidots maps uses the *lat* and *lng* keys from a dot's **context** to extract the coordinates of your device, in that way you just need to send a single dot with the coordinates values in the variable context to plot a map instead of sending separately both latitude and longitude in two different variables. Below you can find a typical **context** with coordinates values:

```
"context" : {"lat":-6.2, "lng":75.4, "weather" : "sunny"}
```

Time Series

Based on the above, we can illustrate an Ubidots time series like this:

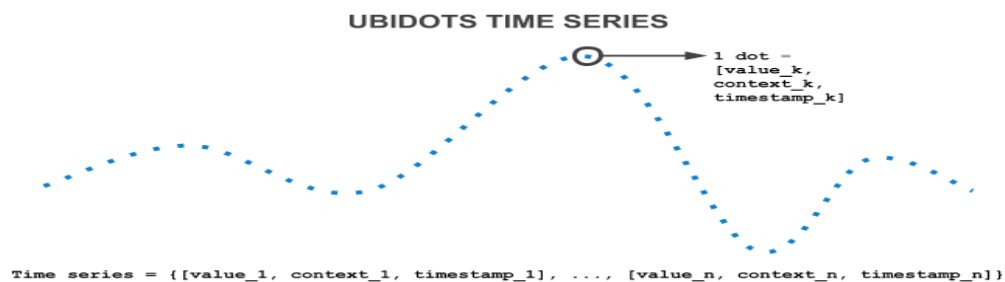


Fig. 4.37: Ubidots time series

Ubidots is an agnostic platform, this means that it does not really care what hardware device you are using, as long as you're able to interact with us through at least one of these protocols:

- HTTP
- MQTT
- TCP/UDP

HTTP

HTTP is the main internet protocol for information exchange in the Internet. It also supports data encryption, which is usually called *secure HTTP* or *HTTPS*. When you use a web browser (MS Edge, Firefox, Google Chrome, etc) you are sending and receiving packages using *HTTP* or *HTTPS*. Because of this, it was not surprising to see *HTTPS* become a popular option for data exchange within IoT.

When a developer is to implement *HTTP* communication, then he should look for a *RESTful Application Programming Interface*, or *REST API*, which exposes all the endpoints required to interact with a third party application (like Ubidots). In this section, you can reference our *REST API* to understand how to send and retrieve data from our servers.

Note: Our goal with this section is to help you understand what's happening behind when communicating with Ubidots, enabling you to replicate it within your own firmware. Because of this, we avoid using our custom libraries in all the examples.

HTTP requests

The following methods are specified within the *HTTP* standard:

HTTP Method	Description
GET	Used to retrieve information
POST	Used to create a new element
PATCH	Used to update existing elements
DELETE	Used to delete existing elements

HTTP is a **request/response** protocol, this means that every request that you make is answered by the server. This response includes a number (response code) and a body. For example, when you make a request to retrieve a file on a

webpage "(e.g. "Get me the file 'websiteside.html')", you build a GET request. If the request is correct, the server will typically return a 200 response code, along with the file requested (body).

Under the same logic, when a device sends a GET request to the Ubidots' server (i.e. requesting the last value of a variable), then the server sends back a response with the status of the request (response code), and a body, which would be the a value with its context and timestamp.

An *HTTP* request also needs the parameters below:

- **Host:** Specifies the server you will be making HTTP requests to.
- **Path:** This is typically the remaining portion of the URL that specifies the resource you want to consume, be it a variable or a device. For example, if an API endpoint is: industrial.api.ubidots.com/api/v1.6/devices/my-device then the path would be /api/v1.6/devices/my-device
- **Headers:** Define the operating parameters of the *HTTP* request such as authentication, Content-Type, Content-Length, etc.
- **Body/payload:** In the case of POST and PATCH requests, this is the data sent by your device to the server. GET requests typically do not have a body because they are meant to request data, not to send data.

Ubidots accepts data as *JavaScript Object Notation* or *JSON*. *JSON* is a typical *HTTP* data type, it is a collection of name/value pairs. In various programming languages, this is treated as an object, record, struct, dictionary, hash table, keyed list, or associative array. It is also human readable and language independent. An example of a *JSON* data type that Ubidots accepts can be referenced below:

```
{"temperature": {"value":10, "timestamp": 1534881387000, "context": {"machine": "1st floor"}}
```

A typical *HTTP* request to Ubidots should be set as below:

```
POST {PATH} HTTP/1.1<CR><LN>
```

Host: {HOST}<CR><LN>

User-Agent: {USER_AGENT}<CR><LN>

X-Auth-Token: {TOKEN}<CR><LN>

Content-Type: application/json<CR><LN>

Content-Length: {PAYLOAD_LENGTH}<CR><LN><CR><LN>

{PAYLOAD}

<CR><LN>

Where:

- {PATH}: Path to the resource to consume
Example: /api/v1.6/variables/ to get user's variables information.
- {HOST}: Host URL.
Example: industrial.api.ubidots.com
- {USER_AGENT}: An optional string used to identify the type of client, be it by application type, operating system, software vendor or software version of the requesting user agent.
Examples: ESP8266/1.0, Particle/1.2
- {TOKEN}: Unique key that authorizes your device to ingest data inside your Ubidots account.
- {PAYLOAD_LENGTH}: The number of characters of your payload.
Example: The payload {"temperature": 20} will have a content-length of 19.
- {PAYLOAD}: Data to send.
Example: {"temperature": 20}

PRO TIP: The easy way of handle *HTTP* and *HTTPs* requests is by using **cURL**, a command line tool. To learn how to install cURL on Windows, MacOSX or Linux please refer to this guide.

API URLs

API access can be over *HTTP* or secure HTTP, *HTTPs*, using the following endpoints based on your Ubidots Account type.

Security Note: We strongly advise to use HTTPs to make sure your data travels encrypted, avoiding the exposure of your API token and/or sensor data.

HTTP

Ubidots Account	Endpoint	Port
Educational	http://things.ubidots.com	80
Industrial	http://industrial.api.ubidots.com	80

HTTPs

Ubidots Account	Endpoint	Port
Educational	https://things.ubidots.com	443
Industrial	https://industrial.api.ubidots.com	443

Ubidots supports SSL v1.1, TLS v1.2 and v1.3. You can download our root certificates in different formats:

- PEM file: A certificate chain with two root certificates from our certificate authorities (CAs).
- DER file: Same as the PEM file, with an alternative encoding.
- CRT file: Same as the PEM file, with a different extension. Often referred to as .crt, .cert or .cer.

Fixed IPs

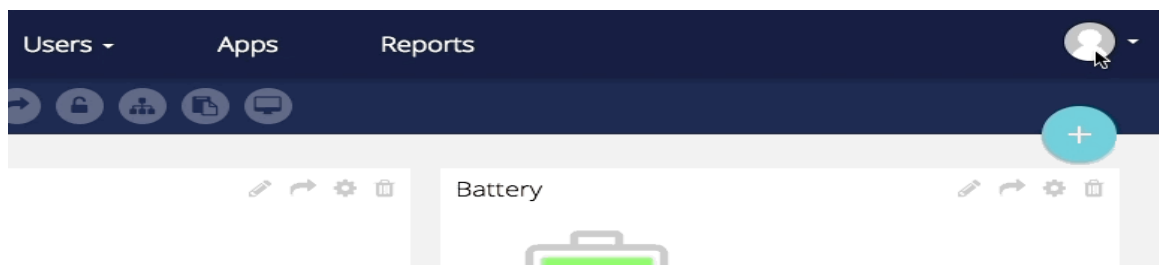
Ubidots also supports fixed IPs address for data ingestion, but we highly advise to our users to implement their firmware routines using the dns endpoints instead of fixed IPs, as these last ones may change in the future.

Ubidots Account	ipv4	ipv6	Port
Educational	50.23.124.68	2607:f0d0:2101:39::2	80/443
Industrial	169.55.61.243	2607:f0d0:3:b::2	80/443

HTTP Authentication

Every request requires a *token*. A *token* is an unique key that authorizes your device to ingest data inside your Ubidots account.

The easiest way to get yours is clicking on "**API Credentials**" under your profile tab:



Also, 6 hour valid *tokens* can be generated using your account *API-KEY*. If you desire to learn how to do it, please refer the Ubidots REST Software API reference.

There are two ways to send a *token* in a request:

1. Send the *token* as parameter in the endpoint, setting it with the `?token=` expression. This is the easiest one to implement but the most unsecure method to send your authentication credentials.
2. Send the *token* as a request header. For this, use the X-Auth-Token header.

Security Note: While sending your *token* as a URL parameter can be quite comfortable in some devices, we only advise to do this for prototyping stages.

For production stages, we strongly advise sending the token using the X-Auth-Token header.

```
GET  
  
https://industrial.api.ubidots.com/api/
```

Send data

Our REST API allows you send data to the platform in two different ways:

- Send data to a **device**
- Send data to a **variable**

If you're looking for more advanced API capabilities, such as editing a device name, or listing all of your devices, please refer to our Ubidots Software REST API Reference.

Send data to a device

Request structure

```
POST /api/v1.6/devices/{DEVICE_LABEL} HTTP/1.1<CR><LN>  
Host: {Host}<CR><LN>  
User-Agent: {USER_AGENT}<CR><LN>  
X-Auth-Token: {TOKEN}<CR><LN>  
Content-Type: application/json<CR><LN>  
Content-Length: {PAYLOAD_LENGTH}<CR><LN><CR><LN>  
{PAYLOAD}  
<CR><LN>
```

Expected Response

```
HTTP/1.1 200 OK<CR><LN>  
Server: nginx<CR><LN>  
Date: Tue, 04 Sep 2018 22:35:06 GMT<CR><LN>
```

```

Content-Type: application/json<CR><LN>
Transfer-Encoding: chunked<CR><LN>
Vary: Cookie<CR><LN>
Allow: GET, POST, HEAD, OPTIONS<CR><LN><CR><LN>
{PAYLOAD_LENGTH_IN_HEXADECIMAL}<CR><LN>
{"{VARIABLE_LABEL}": [{"status_code": 201}]}<CR><LN>
0<CR><LN>

```

The easiest way to **send values to Ubidots** is by specifying your Device label in the request *path* and making a POST request to it.

Replace the {DEVICE_LABEL} with the unique label of the desired device to send data to. If the device does not exist in your Ubidots account, it will be automatically created. Device labels can contain any alphanumerical character, without blank spaces, i.e *weather-station*, *my-home*, *esp8266*.

To set the variables values, just create a *JSON* data payload as below

Nº Variables	JSON structure
One	{"variable-label": VALUE}
Multiple	{"variable-label-1": {"value": VALUE, "timestamp": TIMESTAMP, "context": {"key": VALUE}}, ... , "variable-label-n": {"value": VALUE, "timestamp": TIMESTAMP, "context": {"key": VALUE}}}

timestamp and *context* parameters are optional.

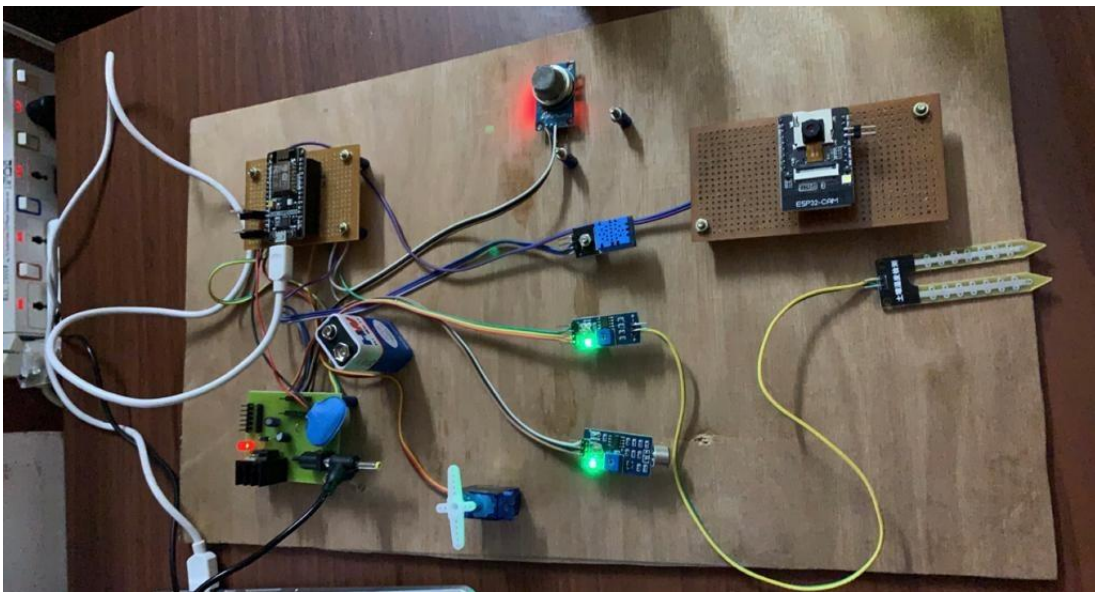
Note 1: By default, the Ubidots backend will take the latitude and longitude values inside any variable labeled as **"gps"**, **"location"**, **position** or **"coordinates"** automatically to get your device coordinates in all of the map widgets. Alternatively, you change the default location variable.

Note 2: All timestamps must be in milliseconds. If you set a timestamp for a variable value, it will be stored with the date represented by it. You can easily convert dates to timestamps values [here](#).

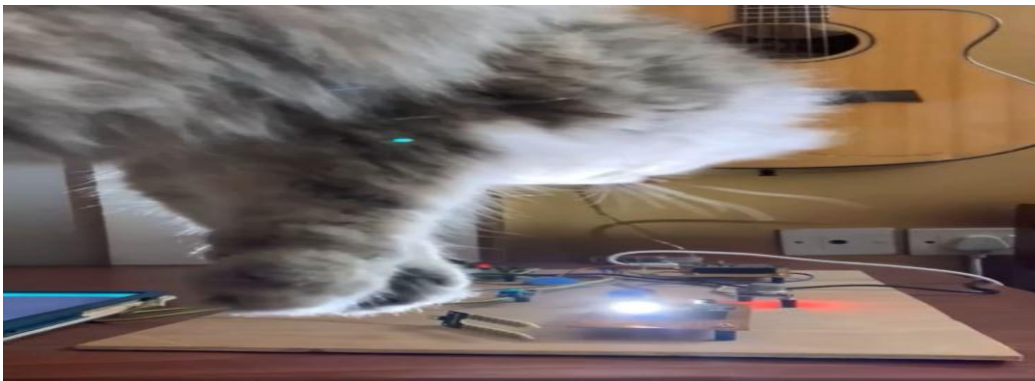
CHAPTER 5

RESULT AND DISCUSSION

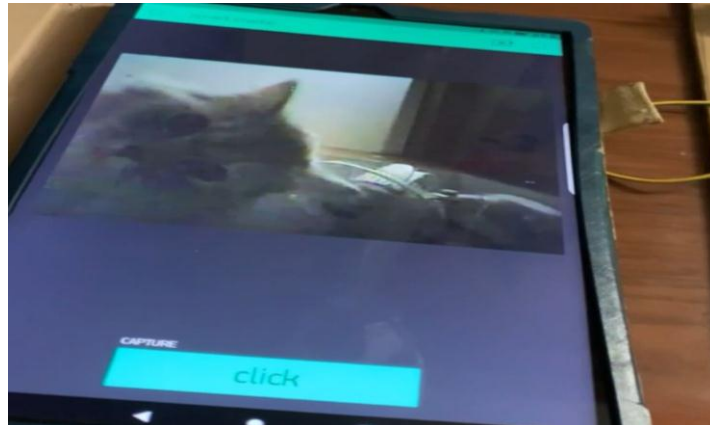
COMPLETE CIRCUIT



CAMERA INPUT



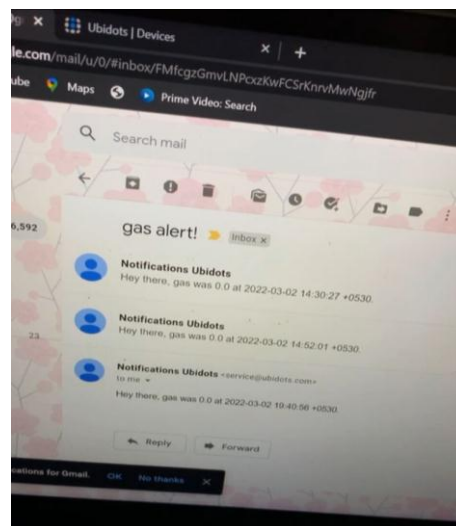
CAMERA OUTPUT



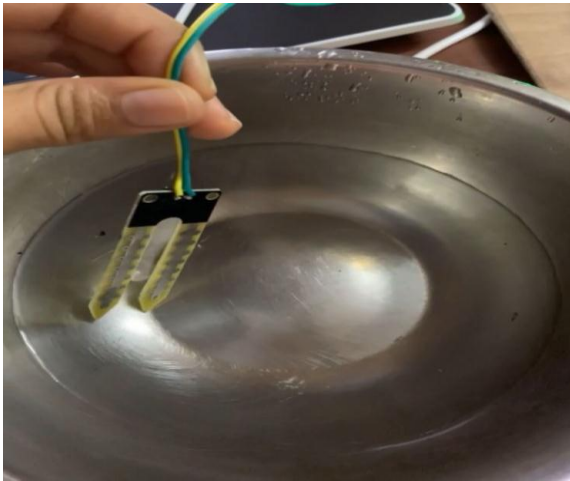
GAS SENSOR INPUT



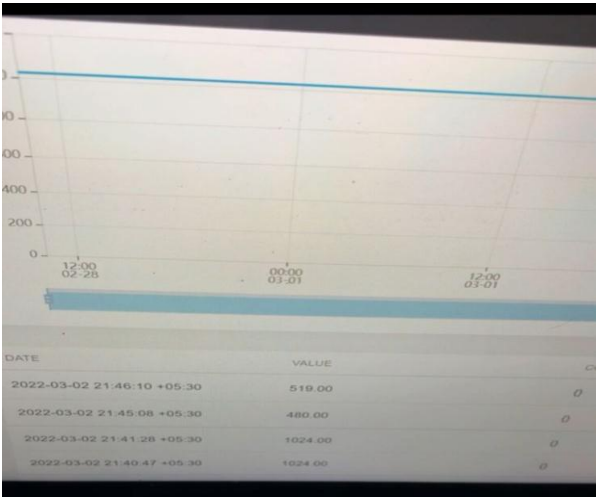
GAS SENSOR OUTPUT



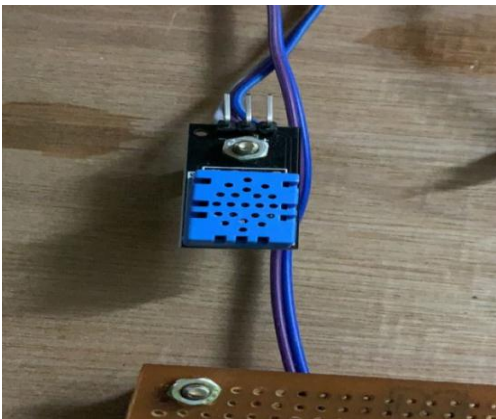
MOISTURE SENSOR INPUT



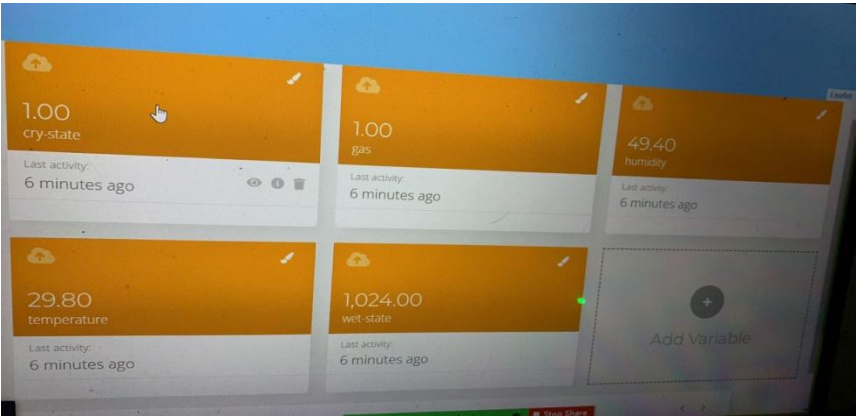
WET STATE OUTPUT



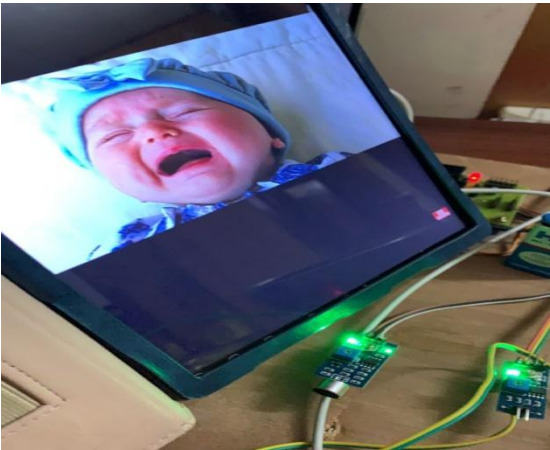
DTH11 SENSOR INPUT



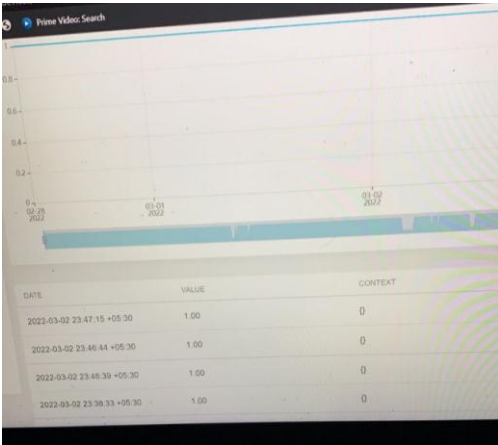
TEMPERATURE AND HUMIDITY READINGS



SOUND SENSOR INPUT



SOUND SENSOR OUTPUT



CHAPTER 6

SUMMARY AND CONCLUSION

Our proposed framework targets observing the indispensable indications of the child, for example, internal heat level utilizing remote innovation and sound sensor used to quantify infant cry's.

We additionally center around increment the extent of sending the data over the web to give far off access.

The camera module consolidated empowers showing the visual feeds of the whereabouts of the child and keeping an eye over their developments in a limited territory. This framework defeats the disadvantage of the current frameworks which are awkward, less easy to use and costly.

REFERENCES

- [1] Anijkar et.al., A.D., "General Idea about Smart Baby Cradle", Int. J. of Innovative Science and Eng., Jan-Feb 2014.
- [2] Aquib Nawaz, "Development of an Intelligent Cradle for Home and Hospital Use", National Inst. of Technology, 2015.
- [3] Harshad Gare, Bhushan Shahane, Kavata Jori, Sweety Jachak, "IOT Based Smart Cradle System for Baby Monitoring", International Research Journal of Engineering and Technology (IRJET), Oct-2019.
- [4] Patil, A. R., "Smart Baby Cradle an IOT based Cradle Management System.", International Conference on Smart City and Emerging Technology (ICSCET), 2018.
- [5] Rachna Palaskar, Shweta Pandey, Ashwini Telang, Akshada Wagh, Ramesh R. Kagalkar, "An Automatic Monitoring and Swing the Baby Cradle for Infant Care" Int. J. of Advanced Research in Computer and Commun. Eng., Dec 2015.