

Marine IoT: non-invasive wearable multisensory platform for oceanic environment monitoring

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Engineering degree in

Electronics and Communication Engineering

By

D.S.K.S AKHIL (Reg No: 37130115)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI-600119

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC 1 12B Status by UGC I Approved by AICTE
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 D.S.K.S AKHIL (RegNo:37130115) who carried out the project entitled “Marine IoT: non-invasive wearable multisensory platform for oceanic environment monitoring” Under my supervision from December 2020 to March 2021.

Internal Guide

DR.M.SUMATHI

Head of the Department

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

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **D.S.K.S AKHIL** (Register No: 37130115) hereby declare that the Project Report entitled “Marine IoT: non-invasive wearable multisensory platform for oceanic environment monitoring” done by us under the guidance of **DR. M. SUMATHI** (ECE) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **ELECTRONICS AND COMMUNICATION ENGINEERING.**

SIGNATURE OF CANDIDATES

D.S.K.S AKHIL (37130115)

DATE:

PLACE: CHENNAI

ACKNOWLEDGEMENT

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

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

We would like to express my sincere and deep sense of gratitude to my Project Guide **DR. M. SUMATHI, ECE.**, for her valuable guidance, suggestions and constant encouragement paved the 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.

ABSTRACTS:

Progresses in the marine research heavily rely on gathering data about species physiology, mobility, and their habitat to understand the environmental changes and its effects using bulky biologgers and sensory networks that are invasive in nature. In this paper, we demonstrate an advanced lightweight compliant environmental monitoring system: “Marine-Skin” with enhanced sensing and logging capabilities, having soft-packaging and endurance up to a depth of 2 km in highly saline Red sea water for multiple weeks. Unlike invasive biologgers, we demonstrate a unique non-invasive attachment strategy by designing a wearable jacket from soft-polymers that can be adhered to any species irrespective of their skin type.

Marine environment monitoring system is of great interest to the researchers since the sea is wealthy store house of natural resources. To Understand and evaluate the marine environment it is essential to collect the data continuously. Several works has been done in the past few decades which employs sophisticated information and communication mechanisms to monitor the marine environment. In this paper, we have proposed a marine environment monitoring system based on under water wireless sensor networks.

Marine environment plays a vital role in life of all living things on this planet. Due to changes in marine environment there is a major setback in the fishery industry, and this setback is escalating every year due to a variety of marine environmental changes. Marine Environment is subjected to over exploitation, eutrophication and introduction of water pollutants by unswerving human activities

Table of contents

S.NO	NAME OF TITLE	PAGE
1	INTRODUCTION	7
2	LITERATURE SURVEY	10
3	AIM AND SCOPE OF THE PRESENT INVESTIGATION	12
4	EMBEDDED SYSTEMS	18
5	RESULTS AND DISCUSSIONS	38
6	CONCLUSIONS AND REFERENCES	41

INTRODUCTION

Monitoring marine environment is forever a tough and fascinating process when taking into accounts its size and expansion. It is very difficult for the human beings to understand the features of the marine environment as they are complex and challenging. Marine environment plays a vital role in life of all living things on this planet. Due to changes in marine environment there is a major setback in the fishery industry, and this setback is escalating every year due to a variety of marine environmental changes. Marine Environment is subjected to over exploitation, eutrophication and introduction of water pollutants by unswerving human activities. Coastline landscape is altered by several natural phenomenons like erosion; these changes can affect progress, scheduling and even security. At present Light Detection and Ranging is used to find out the changes in coastline landscape. Rise in sea level and floods can have severe bang on the life of people. Currently tide stations are used to monitor the rise in sea level. There are a number of incidences by which oil or hazardous chemical are introduced into the water bodies, these spills irrespective of whether it is intentional or unintentional, affects the water animals, fishery industry, ecological diversity and at times it affects human population.

LITERATURE SURVEY:

S.NO	TITLE	AUTHOR	CONTENT
1	Non-invasive featherlight wearable compliant “Marine Skin” – standalone multisensory system for deep-sea environment monitoring	Sohail F. Shaikh ¹ , Harold F. Mazo-Mantilla ¹ , Nadeem Qaiser ¹	Oceanic temperature is relatively stable in response to changes in climates, for example, the change in oceanic temperature is effectively 0.1 °C for 0.6 °C change in global average temperature in the last century. However, the temperature of the seawater varies with the increasing depth (200-1500 m) known as thermocline region, which also has a significant effect on the marine ecosystems at different depths.
2	Marine IoT: non-invasive wearable multisensory platform for oceanic environment monitoring	Sohail Faizan Shaikh, Muhammad Kamran Mustafa	Advances in marine research to understand environmental change and its effect on marine ecosystems rely on gathering data on species physiology, their habitat, and their mobility patterns using heavy and invasive biologgers and sensory telemetric networks. In the past, a lightweight (6 g) compliant environmental monitoring system: Marine Skin was demonstrated. In this paper, an enhanced version of that skin with improved functionalities (500–1500% enhanced sensitivity), packaging, and most importantly its endurance at a depth of 2 km in the highly saline Red Sea water for four consecutive weeks is reported

Marine IoT: non-invasive wearable multisensory platform for oceanic environment monitoring

Abstract:

Marine environment monitoring system is of great interest to the researchers since the sea is wealthy store house of natural resources. To Understand and evaluate the marine environment it is essential to collect the data continuously. Several works has been done in the past few decades which employs sophisticated information and communication mechanisms to monitor the marine environment. In this paper, we have proposed a marine environment monitoring system based on under water wireless sensor networks. Underwater Wireless Sensor Networks (UWSNs) is a structure that performs collaborative duty with the help of different types of nodes .In recent years Underwater Wireless Sensor Networks have been well thought out as capable alternative for monitoring marine environments as they have many advantages such as unmanned function, simple deployment, real-time monitoring and comparatively low expenditure. Our system is designed in such a way that it can process a large amount of data that are collected continuously.

Introduction:

Monitoring marine environment is forever a tough and fascinating process when taking into accounts its size and expansion. It is very difficult for the human beings to understand the features of the marine environment as they are complex and challenging. Marine environment plays a vital role in life of all living things on this planet. Due to changes in marine environment there is a major setback in the fishery industry, and this setback is escalating every year due to a variety of marine environmental changes. Marine Environment is subjected to over exploitation, eutrophication and introduction of water pollutants by unswerving human activities. Coastline landscape is altered by several natural phenomenons like erosion; these changes can affect progress, scheduling and even security. At present Light Detection and Ranging is used to find out the changes in coastline landscape. Rise in sea level and floods can have severe bang on the life of people. Currently tide stations are used to monitor the rise in sea level. There are a number of

incidences by which oil or hazardous chemical are introduced into the water bodies, these spills irrespective of whether it is intentional or unintentional, affects the water animals, fishery industry, ecological diversity and at times it affects human population.

Tsunami which arises in the marine environment cause huge destruction to life and property, sophisticated Real-time seismic monitoring can help us to predict tsunamis. But deep water seismic observations have been meager due to the high cost, hazard, and complexity in collecting data from subsea sensors. Underwater Wireless Sensor Networks (UWSNs) contain a number of mechanisms such as vehicle and sensors that are deployed in a precise location to carry out joint monitoring and data collection responsibilities. These networks are used between different nodes and ground-based stations. Underwater sensor networks nodes are capable of moving from one place to another due to water currents. Since the sensor nodes are mobile the collected data by the nodes would be useful only when localization is concerned. Another key challenge that is affecting underwater sensor networks is energy saving. The energy of the sensor node would be quickly depleted as the sensor nodes are mobile.

1. Non-invasive featherlight wearable compliant “Marine Skin” – standalone multisensory system for deep-sea environment monitoring

Author: Sohail F. Shaikh¹, Harold F. Mazo-Mantilla¹, Nadeem Qaiser¹

Oceanic temperature is relatively stable in response to changes in climates, for example, the change in oceanic temperature is effectively 0.1 °C for 0.6 °C change in global average temperature in the last century. However, the temperature of the seawater varies with the increasing depth (200-1500 m) known as thermocline region, which also has a significant effect on the marine ecosystems at different depths.

2. Marine IoT: non-invasive wearable multisensory platform for oceanic environment monitoring

Author: Sohail Faizan Shaikh, Muhammad Kamran Mustafa

Advances in marine research to understand environmental change and its effect on marine ecosystems rely on gathering data on species physiology,

their habitat, and their mobility patterns using heavy and invasive biologgers and sensory telemetric networks. In the past, a lightweight (6 g) compliant environmental monitoring system: Marine Skin was demonstrated. In this paper, an enhanced version of that skin with improved functionalities (500–1500% enhanced sensitivity), packaging, and most importantly its endurance at a depth of 2 km in the highly saline Red Sea water for four consecutive weeks is reported

11

3. Internet of Things in Marine Environment Monitoring: A Review

Author: Guobao Xu 1 , Yanjun Shi 2 , Xueyan Sun 3 and Weiming Shen 3

11

Marine environment monitoring has attracted more and more attention due to the growing concern about climate change. During the past couple of decades, advanced information and communication technologies have been applied to the development of various marine environment monitoring systems. Among others, the Internet of Things (IoT) has been playing an important role in this area. This paper presents a review of the application of the Internet of Things in the field of marine environment monitoring.

26

4. Advances in Smart Environment Monitoring Systems Using IoT and Sensors

Author: Silvia Liberata Ullo and G. R. Sinha

19

Air quality, water pollution, and radiation pollution are major factors that pose genuine challenges in the environment. Suitable monitoring is necessary so that the world can achieve sustainable growth, by maintaining a healthy society. In recent years, the environment monitoring has turned into a smart environment monitoring (SEM) system, with the advances in the internet of things (IoT) and the development of modern sensors. Under this scenario, the present manuscript aims to accomplish a critical review of noteworthy contributions and research studies on SEM, that involve monitoring of air quality, water quality, radiation pollution, and agriculture systems.

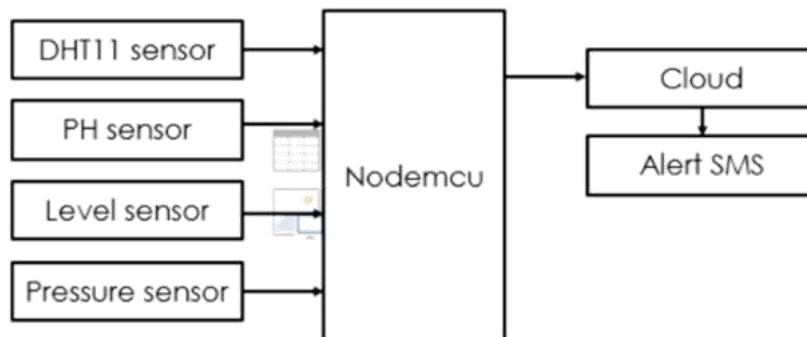
Existing system:

11 Marine environment monitoring has attracted more and more attention due to the growing concern about climate change. During the past couple of decades, advanced information and communication technologies have been applied to the development of various marine environment monitoring systems. Among others, the Internet of Things (IoT) has been playing an important role in this area. This paper presents a review of the application of the Internet of Things in the field of marine environment monitoring. New technologies including advanced Big Data analytics and their applications in this area are briefly reviewed. It also discusses key research challenges and opportunities in this area, including the potential application of IoT and Big Data in marine environment protection.

Proposed system:

15 Progresses in the marine research heavily rely on gathering data about species physiology, mobility, and their habitat to understand the environmental changes and its effects using bulky biologgers and sensory networks that are invasive in nature. In this paper, we demonstrate an advanced lightweight compliant environmental monitoring system: "Marine-Skin" with enhanced sensing and logging capabilities, having soft-packaging and endurance up to a depth of 2 km in highly saline Red sea water for multiple weeks. Unlike invasive biologgers, we demonstrate a unique non-invasive attachment strategy by designing a wearable jacket from soft-polymers that can be adhered to any species irrespective of their skin type. We have successfully deployed the feather-light (<0.5 g in air, 3g with jacket) wearable gadget on Wobbegong shark, Seabream, and common Goldfish to demonstrate the non-invasive and pragmatic attachment mechanism for different species of variable sizes without any hindrance to natural movement or behavior of animal tagged.

Block diagram:



1. EMBEDDED SYSTEMS

9

1.1 Overview of embedded systems

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions, often with real-time computing constraints. It is usually embedded as part of a complete device including hardware and mechanical parts. In contrast, a general-purpose computer, such as a personal computer, can do many different tasks depending on programming. Embedded systems have become very important today as they control many of the common devices we use.

Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product, or increasing the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Physically, embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

In general, "embedded system" is not an exactly defined term, as many systems have some element of programmability. For example, Handheld computers share some elements with embedded systems — such as the operating systems and microprocessors which power them — but are not truly embedded systems, because they allow different applications to be loaded and peripherals to be connected.

Embedded systems provide several functions

- Monitor the environment; embedded systems read data from input sensors. This data is then processed and the results displayed in some format to a user or users
- Control the environment; embedded systems generate and transmit commands for actuators.
- Transform the information; embedded systems transform the data collected in some meaningful way, such as data compression/decompression

Although interaction with the external world via sensors and actuators is an important aspect of embedded systems, these systems also provide functionality specific to their applications. Embedded systems typically execute applications such as control laws, finite state machines, and signal processing algorithms. These systems must also detect and react to faults in both the internal computing environment as well as the surrounding electromechanical systems.

There are many categories of embedded systems, from communication devices to home appliances to control systems. Examples include;

- Communication devices

7

e. g.: modems, cellular phones

- Home Appliances

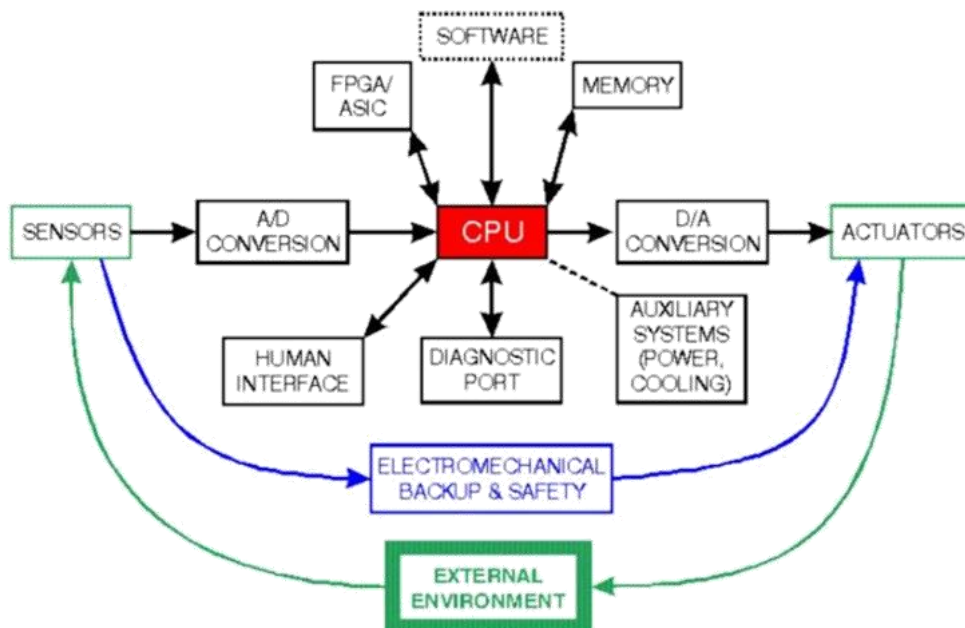
e. g.: CD player, VCR, microwave oven

- Control Systems

e. g.: Automobile anti-lock braking systems, robotics, satellite control

1.2 Block diagram of an embedded system:

An embedded system usually contains an embedded processor. Many appliances that have a digital interface -- microwaves, VCRs, cars -- utilize embedded systems. Some embedded systems include an operating system. Others are very specialized resulting in the entire logic being implemented as a single program. These systems are embedded into some device for some specific purpose other than to provide general purpose computing . A typical embedded system is shown in Fig 1.1



7
Fig 1.1 Block diagram of a typical embedded system

1.2.1 Characteristics of embedded systems

Embedded systems are characterized by a unique set of characteristics. Each of these characteristics imposed a specific set of design constraints on embedded systems designers. The challenge to designing embedded systems is to conform to the specific set of constraints for the application.

Application Specific Systems:

Embedded systems are not general-purpose computers. Embedded system designs are optimized for a specific application. Many of the job characteristics are known before the hardware is designed. This allows the designer to focus on the specific design constraints of a well-defined application. As such, there is limited user reprogram ability. Some embedded systems, however, require the flexibility of reprogram ability. Programmable DSPs are common for such applications.

Reactive Systems

As mentioned earlier, a typical embedded systems model responds to the environment via sensors and control the environment using actuators. This requires embedded systems to run at the speed of the environment. This characteristic of embedded system is called "reactive". Reactive computation means that the system (primarily the software component) executes in response to external events. External events can be either periodic or aperiodic. Periodic events make it easier to schedule processing to guarantee performance. Aperiodic events are harder to schedule. The maximum event arrival rate must be estimated in order to accommodate worst case situations. Most embedded systems have a significant reactive component. One of the biggest challenges for embedded system designers is performing an accurate worst case design analysis on systems with statistical performance characteristics (e.g., cache memory on a DSP or other embedded processor). Real time system operation means that the correctness of a computation depends, in part, on the time at which it is delivered. Systems with this requirement must often design to worst case performance. But accurately predicting the worst case may be difficult on complicated architectures. This often leads to overly pessimistic estimates erring on the side of caution. Many embedded systems have a significant requirement for real time operation in

order to meet external I/O and control stability requirements. Many real-time systems are also reactive systems.

Distributed Systems

A common characteristic of an embedded system is one that consists of communicating processes executing on several CPUs or ASICs which are connected by communication links. The reason for this is economy. Economical 4 8-bit microcontrollers may be cheaper than a 32-bit processors. Even after adding the cost of the communication links, this approach may be preferable. In this approach, multiple processors are usually required to handle multiple time-critical tasks. Devices under control of embedded systems may also be physically distributed.

Heterogeneous Architectures

Embedded systems often are composed of heterogeneous architectures (Fig 1.2). They may contain different processors in the same system solution. They may also be mixed signal systems. The combination of I/O interfaces, local and remote memories, and sensors and actuators makes embedded system design truly unique. Embedded systems also have tight design constraints, and heterogeneity provides better design flexibility.

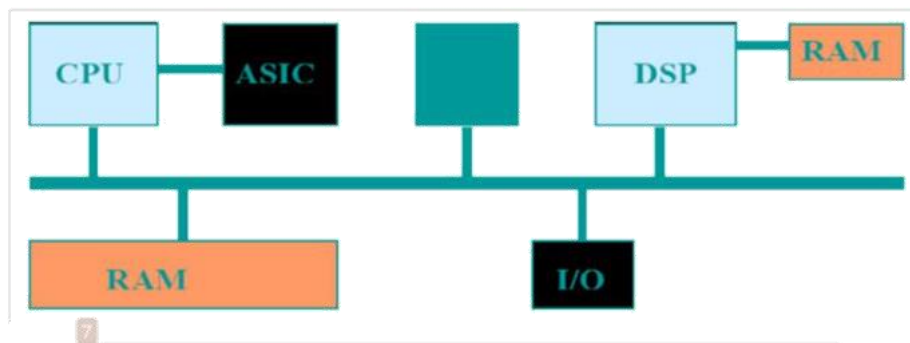


Fig 1.2 Embedded Systems having Heterogeneous Architectures

Harsh environment

Many embedded systems do not operate in a controlled environment. Excessive heat is often a problem, especially in applications involving combustion (e.g., many transportation applications). Additional problems can be caused for embedded computing by a need for protection from vibration,

shock, lightning, power supply fluctuations, water, corrosion, fire, and general physical abuse.

5

System safety and reliability

As embedded system complexity and computing power continue to grow, they are starting to control more and more of the safety aspects of the overall system. These safety measures may be in the form of software as well as hardware control. Mechanical safety backups are normally activated when the computer system loses control in order to safely shut down system operation. Software safety and reliability is a bigger issue. Software doesn't normally "break" in the sense of hardware. However software may be so complex that a set of unexpected circumstances can cause software failures leading to unsafe situations. Discussion of this topic is outside the scope of this book, but the challenges for embedded designers include designing reliable software and building cheap, available systems using unreliable components. The main challenge for embedded system designers is to obtain low-cost reliability with minimal redundancy.

Control of physical systems

One of the main reasons for embedding a computer is to interact with the environment. This is often done by monitoring and controlling external machinery. Embedded computers transform the analog signals from sensors into digital form for processing. Outputs must be transformed back to analog signal levels. When controlling physical equipment, large current loads may need to be switched in order to operate motors and other actuators. To meet these needs, embedded systems may need large computer circuit boards with many non-digital components. Embedded system designers must carefully balance system tradeoffs among analog components, power, mechanical, network, and digital hardware with corresponding software.

Small and low weight

Many embedded computers are physically located within some larger system. The form factor for the embedded system may be dictated by aesthetics. For example, the form factor for a missile may have to fit inside the nose of the missile. One of the challenges for embedded systems designers is to develop

non-rectangular geometries for certain solutions. Weight can also be a critical constraint. Embedded automobile control systems, for example, must be light weight for fuel economy. Portable CD players must be light weight for portability purposes.

Cost sensitivity

Cost is an issue in most systems, but the sensitivity to cost changes can vary dramatically in embedded systems. This is mainly due to the effect of computer costs have on profitability and is more a function of the proportion of cost changes compared to the total system cost.

Power management

Embedded systems have strict constraints on power. Given the portability requirements of many embedded systems, the need to conserve power is important to maintain battery life as long as possible. Minimization of heat production is another obvious concern for embedded systems.

1.3 POWER SUPPLY

All electronic circuits works only in low DC voltage, so we need a power supply unit to provide the appropriate voltage supply for their proper functioning .This unit consists of transformer, rectifier, filter & regulator. AC voltage of typically 230volts rms is connected to a transformer voltage down to the level to the desired ac voltage. A diode rectifier that provides the full wave rectified voltage that is initially filtered by a simple capacitor filter to produce a dc voltage. This resulting dc voltage usually has some ripple or ac voltage variation . A regulator circuit can use this dc input to provide dc voltage that not only has much less ripple voltage but also remains the same dc value even the dc voltage varies some what, or the load connected to the output dc voltages changes.

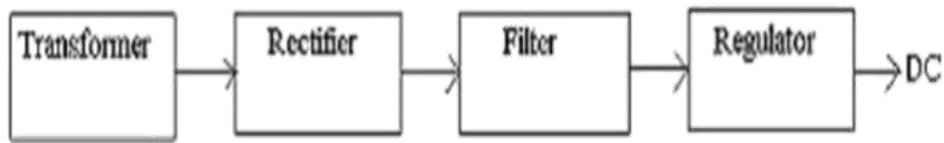


Fig 1. General Block Of Power Supply Unit

1.4 TRANSFORMER:

A transformer is a static piece of which electric power in one circuit is transformed into electric power of same frequency in another circuit. It can raise or lower the voltage in the circuit, but with a corresponding decrease or increase in current. It works with the principle of mutual induction. In our project we are using a step down transformer to providing a necessary supply for the electronic circuits. Here we step down a 230volts ac into 12volts ac.

1.5 RECTIFIER:

A dc level obtained from a sinusoidal input can be improved 100% using a process called full wave rectification. Here in our project for full wave rectification we use bridge rectifier. From the basic bridge configuration we see that two diodes(say D2 & D3) are conducting while the other two diodes (D1 & D4) are in off state during the period $t = 0$ to $T/2$. Accordingly for the negative cycle of the input the conducting diodes are D1 & D4 .Thus the polarity across the load is the same.

In the bridge rectifier the diodes may be of variable types like 1N4001, 1N4003, 1N4004, 1N4005, 1N4007 etc.... can be used. But here we use 1N4007, because it can withstand up to 1000v.

1.6 FILTERS:

In order to obtain a dc voltage of 0 Hz, we have to use a low pass filter. So that a capacitive filter circuit is used where a capacitor is connected at the rectifier output & a dc is obtained across it. The filtered waveform is essentially a dc voltage with negligible ripples & it is ultimately fed to the load.

1.7 REGISTERS:

The controller IC has two 8 bit registers, an instruction register (IR) and a data register (DR). The IR stores the instruction codes and address information for display data RAM (DD RAM) and character generator RAM (CG RAM). The IR can be written, but not read by the MPU.

The DR temporally stores data to be written to /read from the DD RAM or CG RAM. The data written to DR by the MPU, is automatically written to the DD RAM or CG RAM as an internal operation.

When an address code is written to IR, the data is automatically transferred from the DD RAM or CG RAM to the DR. data transfer between the MPU is then completed when the MPU reads the DR. likewise, for the next MPU read of the DR, data in DD RAM or CG RAM at the address is sent to the DR automatically. Similarly, for the MPU write of the DR, the next DD RAM or CG RAM address is selected for the write operation.

22
The dot-matrix liquid crystal display controller and driver LSI displays alphanumeric, Japanese kana characters, and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8-bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver

1.8 REGULATORS:

18
The output voltage from the capacitor is more filtered & finally regulated. The voltage regulator is a device, which maintains the output voltage constant irrespective of the change in supply variations, load variations & temperature changes. Here we use fixed voltage regulator namely LM7805. The IC LM7805 is a +5v regulator which is used for microcontroller.

24 1.9 FEATURES AND DESCRIPTION OF REGULATORS

- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

10 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. 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. Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the [gerber](#) file of an ESP8266 board, named devkit v0.9. Later that month, Tuan PM ported [MQTT](#) client library from [Contiki](#) to the ESP8266 SoC platform, and committed to NodeMCU project, then NodeMCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to NodeMCU project, enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays.

In summer 2015 the creators abandoned the firmware project and a group of independent contributors took over. By summer 2016 the NodeMCU included more than 40 different modules. Due to resource constraints users need to select the modules relevant for their project and build a firmware tailored to their needs.

[8](#)

ESP8266 Arduino Core [\[edit\]](#)

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. Some ESP8266 enthusiasts developed an Arduino core for the ESP8266 WiFi SoC, popularly called the "ESP8266 Core for the Arduino IDE".^[16] This has become a leading software development platform for the various ESP8266-based modules and development boards, including NodeMCUs.

6 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.



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.



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.

There is online NodeMCU custom builds available using which we can easily get our custom NodeMCU firmware 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](#)

Difference in using ESPlorer and Arduino IDE

Well, there is a programming language difference we can say while developing application for NodeMCU using ESPlorer IDE and Arduino IDE.

We need to code in C\C++ programming language if we are using Arduino IDE for developing NodeMCU applications and Lua language if we are using ESPlorer IDE.

Basically, NodeMCU is Lua Interpreter, so it can understand Lua script easily. When we write Lua scripts for NodeMCU and send/upload it to NodeMCU, then they will get executes sequentially. It will not build binary firmware file

of code for NodeMCU to write. It will send Lua script as it is to NodeMCU to get execute.

In Arduino IDE when we write and compile code, ESP8266 toolchain in background creates binary firmware file of code we wrote. And when we upload it to NodeMCU then it will flash all NodeMCU firmware with newly generated binary firmware code. In fact, it writes the complete firmware.

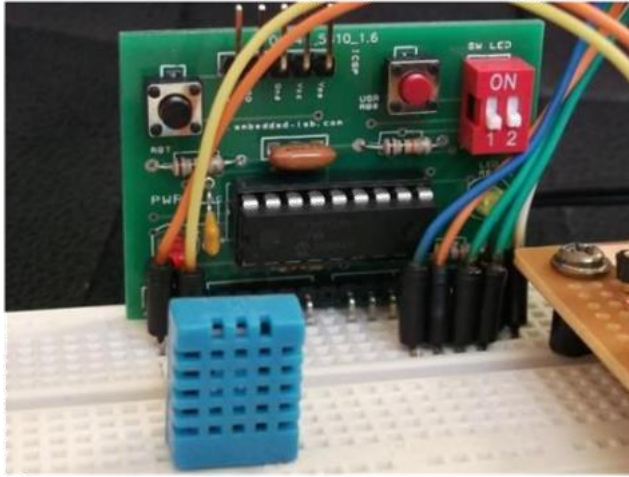
That's the reason why NodeMCU not accept further Lua scripts/code after it is getting flashed by Arduino IDE. After getting flashed by Arduino sketch/code it will be no more Lua interpreter and we got error if we try to upload Lua scripts. To again start with Lua script, we need to flash it with NodeMCU firmware.

Since Arduino IDE compile and upload/writes complete firmware, it takes more time than ESPlorer IDE.

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](#). These sensors are capable of measuring both temperature and relative humidity and provide fully calibrated digital outputs. While SHT1x/SHT7x are very accurate sensors, they are still expensive for hobbyists use. This articles discusses the DHT11 sensor which also provides calibrated digital outputs for temperature and humidity but is relatively lot cheaper than the Sensirion 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

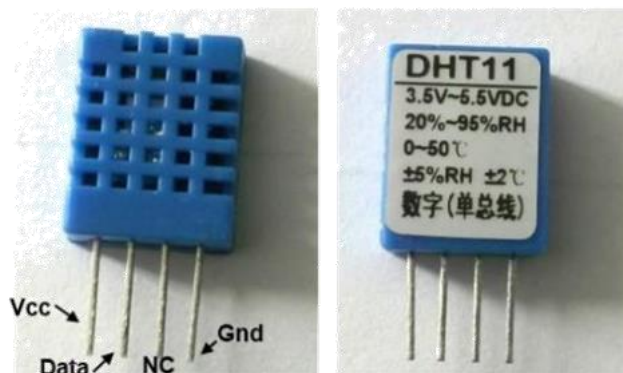
temperature and humidity values from the sensor and display them on a 16×2 character LCD.



Interfacing DHT11 sensor with PIC16F628A

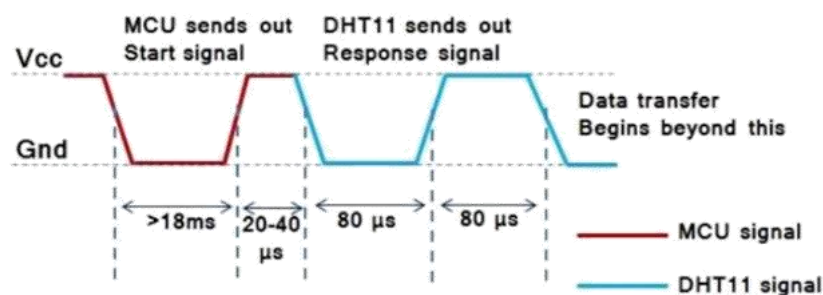
About DHT11 sensor

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.



DHT11 sensor comes in a single row 4-pin package

The following timing diagrams describe the data transfer protocol between a MCU and the DHT11 sensor. The MCU initiates data transmission by issuing a **"Start"** signal. The MCU pin must be configured as output for this purpose. The MCU first pulls the data line low for at least 18 ms and then pulls it high for next 20-40 μ s before it releases it. Next, the sensor responds to the MCU **"Start"** signal by pulling the line low for 80 μ s followed by a logic high signal that also lasts for 80 μ s. Remember that the MCU pin must be configured to input after finishing the **"Start"** signal. Once detecting the response signal from the sensor, the MCU should be ready to receive data from the sensor. The sensor then sends 40 bits (5 bytes) of data continuously in the data line. Note that while transmitting bytes, the sensor sends the most significant bit first.



"Start" and "Response" signals

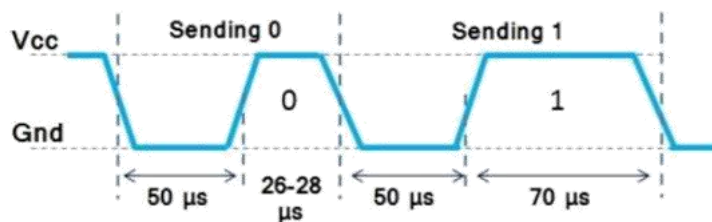
The 40-bit data from the sensor has the following structure.

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

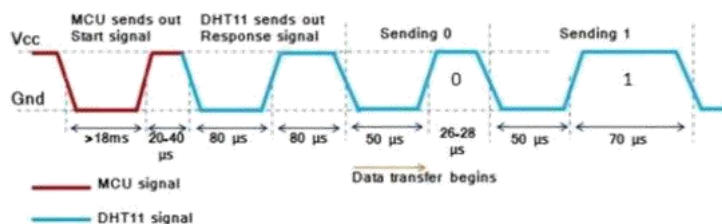
For DHT11 sensor, the decimal bytes of temperature and humidity measurements are always zero. Therefore, the first and third bytes of received data actually give the numeric values of the measured relative humidity (%) and temperature ($^{\circ}$ C). The last byte is the checksum byte which is used to 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"



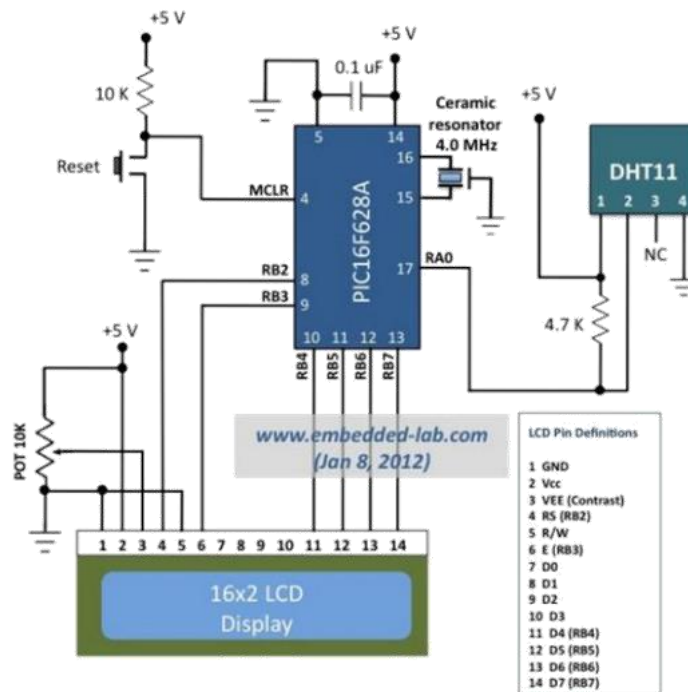
Start, Response and Data signals in sequence

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

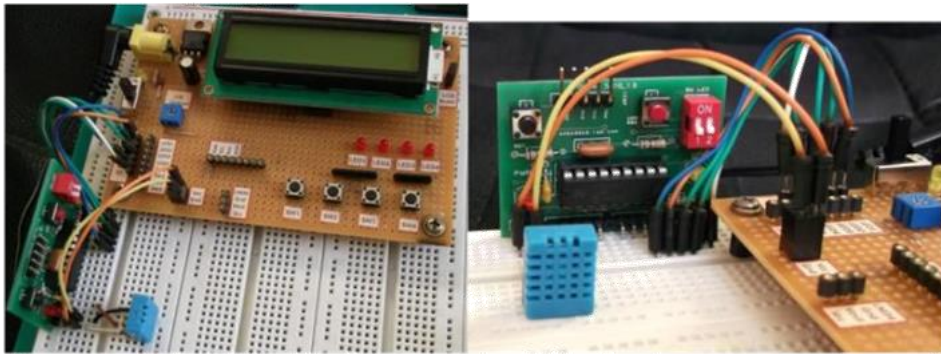
Here is the circuit diagram showing the DHT11 sensor and a HD44780-based character LCD interfaced to the PIC16F628A microcontroller. The

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 μ s. 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.



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.



Complete setup of the circuit

PIC16F628A module and the DHT11 sensor are plugged into the breadboard

Software

Writing a software for DHT11 sensor is little more challenging than the hardware part because of the timing conditions for 1s and 0s. I have written sub-routines in mikroC Pro for PIC for initializing the DHT11 sensor and reading the 40-bit of data in sequence. I have used Timer2 module to keep track of the width of the received data pulse, which is required to identify if the received bit is 1 or 0. When a low-to-high pulse is detected at the beginning of any data bit, TMR2 is cleared and turned ON. Since the clock frequency used here is 4.0 MHz, the TMR2 increments by 1 in every 1 μ s. The TMR2 is stopped whenever the data pulse is low again. The value of the TMR2 register gives you the width of the data pulse in μ s. I am using 40 μ s as the threshold for identifying 0 and 1. If the TMR2 is greater than 40, it means the received bit is 1, else it is 0. Here is the complete source code written in mikroC Pro for PIC. It can be easily adapted to any other platform, but remember that if you are using a different clock frequency you should have to modify the timer operation accordingly.

BMP 180 sensor

A pressure sensor measures pressure, typically of gases or liquids. Pressure is an expression of the force required to stop a fluid from expanding, and is usually stated in terms of force per unit area. A pressure

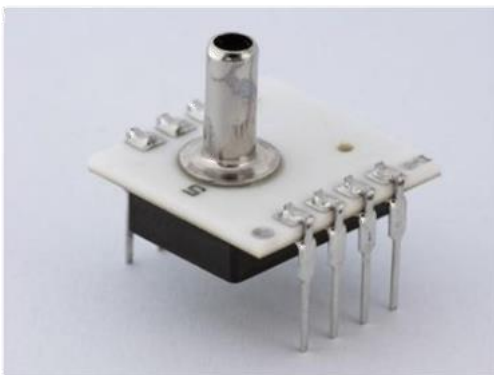
sensor usually acts as a transducer; it generates a signal as a function of the pressure imposed. For the purposes of this article, such a signal is electrical.

Pressure sensors are used for control and monitoring in thousands of everyday applications. Pressure sensors can also be used to indirectly measure other variables such as fluid/gas flow, speed, water level, and altitude. Pressure sensors can alternatively be called pressure transducers, pressure transmitters, pressure senders, pressure indicators, piezometers and manometers, among other names.

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

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

Some pressure sensors, such as those found in some traffic enforcement cameras, function in a binary (off/on) manner, i.e., when pressure is applied to a pressure sensor, the sensor acts to complete or break an electrical circuit. These types of sensors are also known as a pressure switch.



Conclusion:**Result:****References:**

- [1] J. B. C. Jackson et al., "Historical overfishing and the recent collapse of coastal ecosystems," *Science*, vol. 293, no. 5530. American Association for the Advancement of Science, pp. 629–637, 27-Jul2001.
- [2] Y. Yuan et al., "Ultra-high mobility transparent organic thin film transistors grown by an off-centre spin-coating method," *Nat. Commun.*, vol. 5, pp. 3327–3334, Jan. 2014
- [3] C. M. Duarte, "Global change and the future ocean: a grand challenge for marine sciences," *Front. Mar. Sci.*, vol. 1, p. 63, Dec. 2014.
- [4] B. S. Halpern et al., "A global map of human impact on marine ecosystems," *Science*, vol. 319, no. 5865, pp. 948–52, Feb. 2008.

- [5] C. J. Bridger and R. K. Booth, "The Effects of Biotelemetry Transmitter Presence and Attachment Procedures on Fish Physiology and Behavior," *Rev. Fish. Sci.*, vol. 11, no. 1, pp. 13–34, Jan. 2003.
- [6] N. Jepsen, C. Schreck, S. Clements, and E. B. Thorstad, "Aquatic Telemetry: Advances and Applications : Proceedings of the Fifth ... - Food and Agriculture Organization of the United Nations - Google Books," *Aquatic Telemetry: Advances and Applications: Proceedings of the Fifth Conference on Fish Telemetry*, pp. 255–260, 2003.
- [7] C. C. Wilmers, B. Nickel, C. M. Bryce, J. A. Smith, R. E. Wheat, and V. Yovovich, "The golden age of bio-logging: how animal-borne sensors are advancing the frontiers of ecology," *Ecology*, vol. 96, no. 7, pp. 1741–1753, Jul. 2015.
- [8] T. Todd Jones et al., "Calculating the ecological impacts of animalborne instruments on aquatic organisms," *Methods Ecol. Evol.*, vol. 4, no. 12, pp. 1178–1186, 2013.
- [9] S. F. Shaikh, M. T. Ghoneim, G. A. Torres Sevilla, J. M. Nassar, A. M. Hussain, and M. M. Hussain, "Freeform Compliant CMOS Electronic Systems for Internet of Everything Applications," *IEEE Trans. Electron Devices*, vol. 64, no. 5, pp. 1894–1905, May 2017.

17

LEVEL MEASUREMENT

In industry, liquids such as water, chemicals, and solvents are used in various processes. The amount of such liquid stored can be found by measuring level of the liquid in a container or vessel. The level affects not only the quantity delivered but also pressure and rate of flow in and out of the container. Level sensors detect the level of substances like liquids, slurries, granular materials, and powders. The substance to be measured can be inside a container or can be in its natural form (e.g. a river or a lake). The level measurement can be either continuous or point values.

18

Wide spectrum of sensors is available in the market and commonly, they are classified based on the specific application of the sensor. Sensor used for measuring humidity is termed as humidity sensor, the one used for measurement of pressure is called pressure sensor, sensor used for measurement of displacement is called position sensor and so on though all of them may be using the similar sensing principle. In a similar fashion, the sensor used for measurement of fluid levels is called a level sensor.

Quite obvious from its name, level sensors are used to measure the level of the free-flowing substances. Such substances include liquids like water, oil, slurries, etc as well as solids in granular/powder form (solids which can flow). These substances tend to get settled in the container tanks due to gravity and maintain their level in rest state. Level sensors measure their level against a pre-set reference.

Point-level sensors indicate whether the substance is above or below the sensing point. This is essential to avoid overflow or emptying of tanks and to protect pumps from dry run.

¹² From the application point of view the considerations are:

- Price
- Accuracy
- Response rate
- Ease of calibration
- Physical size and mounting of the instrument
- Monitoring or control of continuous or discrete levels

Level measurements are broadly classified in two groups:

- Direct methods
- Indirect methods

In direct methods, the level is indicated directly by means of simple mechanical devices. The measurement is not affected by changes in material density. Few examples are:

- Dip Stick
- Resistance Tapes
- Sight Glass
- Floats
- Ultrasonic
- Radar

In Indirect methods, the level is converted in a measurable signal using a suitable transducer. Change in the material affects the measurement. A corrective factor must be used in recalibrating the instrument. Few examples are:

- Hydrostatic head methods
- Load cell

- Capacitance
- Conductivity

Applications:

¹²
Level Probes are used for measuring level of

1. Liquids
2. Powdered and granular solids
3. Liquid metals at very high temperature
4. Liquefied gases at very low temperature

¹²
Disadvantages:

Light density materials under 20 lb/ft³ and materials with particle sizes exceeding 1/2 in. in diameter can be a problem due to their very low dielectric constants (caused by the large amount of air space between particles).

IOT:

³
The structure of this chapter is the following: Section 1.1 defines the motivation for this work. Section 1.2 shows the research objectives and questions. Section 1.3 presents the steps required to achieve our objectives. Section 1.4 shows the structure of the report.

1.1 Background

The Future Internet goal is to provide an infrastructure to have an immediate access to information about the physical world and its objects. Physical objects can be applicable to different application domains, such as e-health, warehouse management, etc. Each application domain may have different

types of physical devices. Each physical device can have its own specifications, which is required to use in order to interact with it. To achieve the future Internet goal, a layered vision is required that can facilitate data access. Internet of Things (IoT) is a vision that aims to integrate the virtual world of information to the real world of devices through a layered architecture.

The term "Internet of Things" consists of two words, namely *Internet* and *Things*. *Internet* refers to the global network infrastructure with scalable, configurable capabilities based on interoperable and standard communication protocols. *Things* are physical objects or devices, or virtual objects, devices or information, which have identities, physical attributes and virtual personalities, and use intelligent interfaces [1]. For instance, a virtual object can represent an abstract unit of sensor nodes that contains metadata to identify and discover its corresponding sensor nodes. Therefore, IoT refers to the *things* that can provide information from the physical environment through the Internet.

Middleware is as an interface between the hardware layer and the application layer, which is responsible for interacting with devices and information management [2]. The role of a middleware is to present a unified programming model to interact with devices. A middleware is in charge of masking the heterogeneity and distribution problems that we face when interacting with devices [3].

1.2 Motivation

IoT-based system is in charge of providing knowledge from an environment to an non-expert user. IoT-based system can be used in different environments, so it needs to be able to address many heterogeneous devices. Thus, a major concern within developing an IoT-based system is how to handle the interaction with the heterogeneous devices for non-expert users. This concern can be addressed by a middleware layer between devices and non-expert users. This layer is responsible to hide the diversity of devices from the user perspective, and provides access transparency to the devices for the end users.

```

<dpws:ThisModel>
  <dpws:Manufacturer>ACME Manufacturing</dpws:Manufacturer>
  <dpws:ModelName xml:lang="en-GB" >
    ColourBeam 9
  </dpws:ModelName>
  <dpws:ModelName xml:lang="en-US" >
    ColorBeam 9
  </dpws:ModelName>
</dpws:ThisModel>
</wsx:MetadataSection>
<wsx:MetadataSection
  dialect="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/ThisDevice"
>
  <dpws:ThisDevice>
    <dpws:FriendlyName xml:lang="en-GB" >
      ACME ColourBeam Printer
    </dpws:FriendlyName>
    <dpws:FriendlyName xml:lang="en-US" >
      ACME ColorBeam Printer
    </dpws:FriendlyName>
  </dpws:ThisDevice>

```

Code 2.1 XML code to define a device

2.3.3 Central control, Context detection & Management (CCM)

Context characterizes the situation of an entity, which can be a place, a person or an object that is relevant to the user, applications and their interactions [1]. The CCM functional component is responsible to support context-aware computation that is a computational style that take to account the context of the entities that interact with the system. A middleware for IoT-based systems must be context-aware to work in smart environments [1]. Smart environments refer to a *physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network* [17]. Context-awareness includes two functionalities:

- 1) Context detection, which consists of collecting data from resources, and selecting the information that can have an impact

on the computation.

- 2) Context processing, to use the gathered information to perform a task or make a decision.

2.3.4 Application Abstraction

This functional component provides an interface for both high-level applications and end-users to interact with devices. For instance, this interface can be a RESTful interface or can be implemented with some query-based language.

A. Malatras. et. al. propose a SOA-based middleware to perform data management and data monitoring services [18]. This middleware uses a RESTful interface to facilitate the interaction of high-level applications with sensors, which can communicate with Wireless Sensor Network (WSN) through the HTTP operations: (1) GET to issue a query on an existing resource, (2) DELETE to remove an existing resource, (3) POST to create a new resource, and (4) PUT to update an existing resource. For instance, a client application gets the result of a domain task by sending a GET request through the URL: `http://{hostname}/REST/{version}/DomaintaskResult/id'`. In this URI that reference to an „id“ leads to a unique result.

In the TinyDB middleware [19], an end-user can use a query language to interact with devices. For instance, a user can send the following query to get the id of a sensor (nodeid) nearby and temperatures that are sensed by this sensor during the past 10 seconds before executing the query:

```
SELECT nodeid, temp
FROM sensors
```

```
SAMPLE PERIOD 1s FOR 10s
```

2.4 Conclusion

In this chapter, we defined IoT and identified common IoT layers. Furthermore, we discussed a reference middleware architecture for IoT-based systems. This architecture has been proposed by Bandyopadhyay, S. et. al based on a study on the existing middleware frameworks for IoT-based systems [1].

Chapter 3 Review of IoT-based Middleware

The term *Internet of Things (IoT)* refers to a wide set of applications and research areas such as, distributed computing and knowledge management. Much of the IoT research we found in the literature is related to the field of *pervasive computing*. World Wide Web Consortium (W3C)³ defines *pervasive computing* as a *vision about our future computing life style. In this vision, computer systems will be seamlessly integrated into our everyday life, anticipating our needs and providing relevant services and information for us in an anytime-and-any-where fashion. Pervasive Computing is all about making the human's life easier by exploiting the available computing technologies* [20]. This motivates scientists to develop new technologies for the everyday people's lives.

Since the vision of IoT is almost similar to Pervasive computing, we reviewed the literatures that address both IoT and Pervasive systems. To select the middleware to discuss we had the following approaches:

1) We looked at typical IoT middleware such as ISMP [21], ASPIRE [22] GSN [23]. We selected HYDRA [24] to review, because it is the most popular and well-documented middleware in comparer to the mentioned middleware. Also, based on the study of Bandyopadhyay, S. et. al [3], those middleware does not support the discussed functional components in Chapter 3, while HYDRA supports them.

2) We looked at typical Pervasive system middleware such as [25], [26], [27]. We selected the following middleware to review:

❑ AURA [28] because this middleware focus on elaboration and manipulation of the gathered data from devices. we aim to provide ease of configuration and deployment for end user and developer. Since to meet this goal, we require to facilitate the gathered data manipulation, we reviewed AURA that was the only middleware among the reviewed paper that considered the data manipulation.

❑ TinyDB [19] focus on gathering data from devices. Since in an IoT-based system we need to gather data from environment through different devices, we reviewed TinyDB that is a popular middleware.

❑ WiseMID [29] is the only middleware among the reviewed middleware that is specific for energy saving purpose. As saving the energy of devices is important issue, we reviewed this middleware.

In this chapter, we the mentioned middleware in

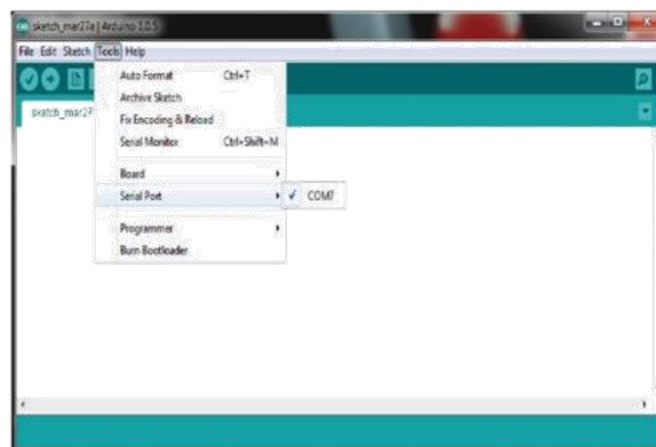
more detail. Also, we analyze the discussed middleware with respect to the functional components mentioned in Chapter 2.

ARDUINO SOFTWARE:

1. Open Arduino IDE as shown below
Open Arduino IDE

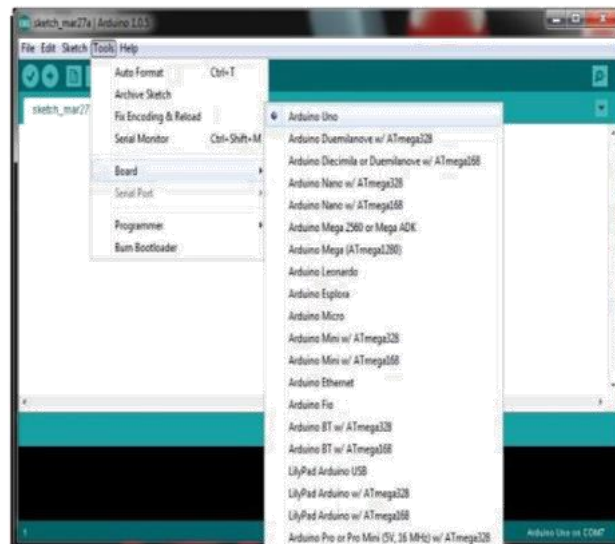


2. Select the COM Port from tool Select the COM Port

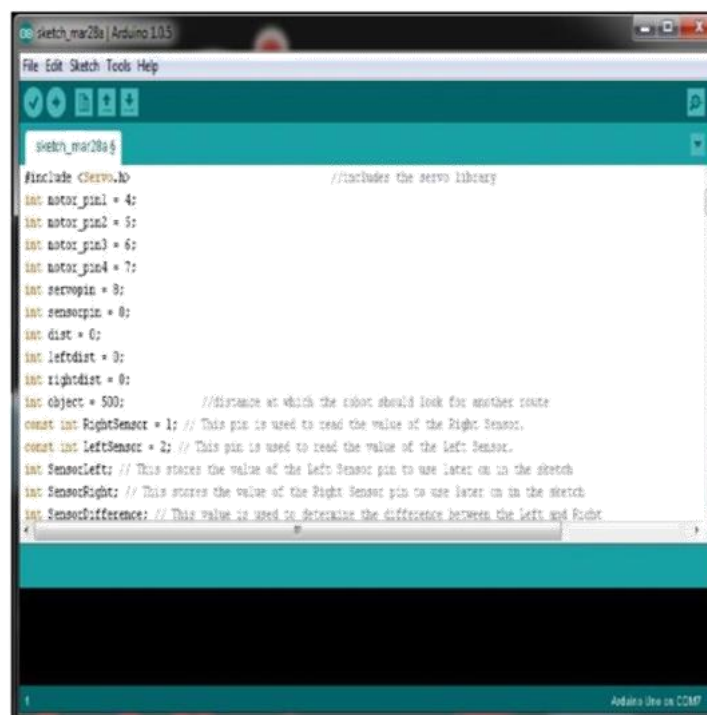


25

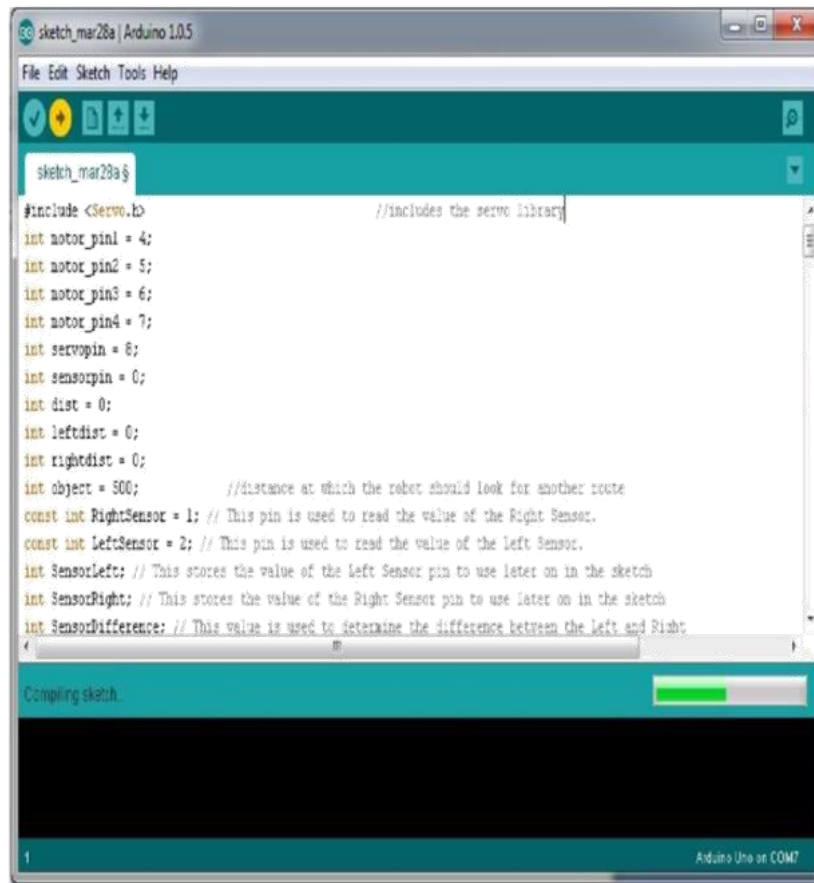
3. Select the required Arduino board from Tools as shown below
Select the required Arduino board



4. Write the sketch in Arduino IDE
Sketch in Arduino IDE



5. Compile and upload the Sketch to Arduino board



6. CONCLUSION

During the past couple of decades, marine environment monitoring has attracted wide attention. Governments and research organizations have heavily invested in the research and development of new technologies in this area. Advanced information and communication technologies have been applied to the development of various marine environment monitoring technologies and systems. Internet of Things has played an important role in this area. This paper presents an updated review of the related technologies and systems on the application of the Internet of Things in marine environment monitoring. A comprehensive review of about 40 related projects revealed that most systems and applications developed thus far are for ocean sensing and monitoring and water quality monitoring. Some specific efforts have been made for fish farm monitoring, coral reef monitoring, wave and current monitoring. Several projects focused on specific technologies or devices like buoys, energy saving and harvesting devices, routing protocols, data transmitting mechanisms, and data analysis techniques. It can also be noted that testing places are very much different among these projects. About half of the developed systems have been tested or deployed in real marine or river environments; a number of them were experimented in lab settings or indoor environments; a few others were tested in outdoor pools or small ponds and lakes; and several of them were only tested through simulations.

While most systems use GPRS and/or ZigBee for wireless communication, a few systems used underwater acoustic communication. Emerging offshore and deep-sea fish farming provides more opportunities and also challenges, not only for marine environment monitoring but also for operation controls and marine environment protection, which is a new area of research. The energy management issue is mainly considered from two aspects: Reducing energy consumption and using alternative renewable energy sources. Optimizing network topologies and developing advanced routing protocols are the ways to reduce energy consumption. Even though many researchers discussed different options for energy harvesting including solar, wind, waves and ocean currents as alternative renewable energy sources, only solar energy has been used in about a third of the developed systems. The results of Big Data analytics can be used not only for feedback to marine environment management agencies and control centers for quick decision making and real time manual interventions but also for autonomous vessels and remotely deployed devices to take real time actions in order to protect the marine environment from some disasters (e.g., oil spills and bad weather). It is a growing area of research and development. This review also identified several research challenges and opportunities, including energy management, standardization of system platforms and technologies, and marine environment protection measures.

REFERENCES

- ▶ [1] J. B. C. Jackson et al., "Historical overfishing and the recent collapse of coastal ecosystems," *Science*, vol. 293, no. 5530. American Association for the Advancement of Science, pp. 629–637, 27-Jul2001.
- ▶ [2] Y. Yuan et al., "Ultra-high mobility transparent organic thin film transistors grown by an off-centre spin-coating method," *Nat. Commun.*, vol. 5, pp. 3327–3334, Jan. 2014
- ▶ [3] C. M. Duarte, "Global change and the future ocean: a grand challenge for marine sciences," *Front. Mar. Sci.*, vol. 1, p. 63, Dec. 2014.
- [4] B. S. Halpern et al., "A global map of human impact on marine ecosystems.," *Science*, vol. 319, no. 5865, pp. 948–52, Feb. 2008

- ▶ [5] C. J. Bridger and R. K. Booth, "The Effects of Biotelemetry Transmitter Presence and Attachment Procedures on Fish Physiology and Behavior," *Rev. Fish. Sci.*, vol. 11, no. 1, pp. 13–34, Jan. 2003.
- ▶ [6] N. Jepsen, C. Schreck, S. Clements, and E. B. Thorstad, "Aquatic Telemetry: Advances and Applications : Proceedings of the Fifth ... - Food and Agriculture Organization of the United Nations - Google Books," *Aquatic Telemetry: Advances and Applications: Proceedings of the Fifth Conference on Fish Telemetry*, pp. 255–260, 2003.
- ▶ [7] C. C. Wilmers, B. Nickel, C. M. Bryce, J. A. Smith, R. E. Wheat, and V. Yovovich, "The golden age of bio-logging: how animal-borne sensors are advancing the frontiers of ecology," *Ecology*, vol. 96, no. 7, pp. 1741–1753, Jul. 2015.
- ▶ [8] T. Todd Jones et al., "Calculating the ecological impacts of animalborne instruments on aquatic organisms," *Methods Ecol. Evol.*, vol. 4, no. 12, pp. 1178–1186, 2013.
- ▶ [9] S. F. Shaikh, M. T. Ghoneim, G. A. Torres Sevilla, J. M. Nassar, A. M. Hussain, and M. M. Hussain, "Freeform Compliant CMOS Electronic Systems for Internet of Everything Applications," *IEEE Trans. Electron Devices*, vol. 64, no. 5, pp. 1894–1905, May 2017.

