

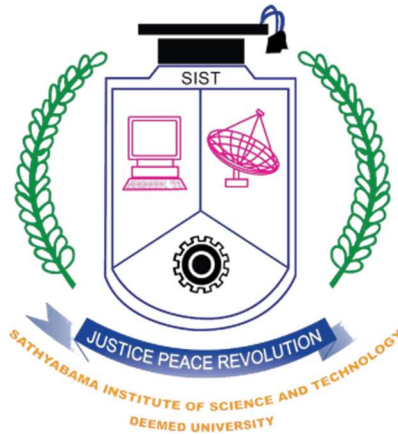
AUTOMATED ANESTHESIA CONTROL SYSTEM AND PATIENT MONITORING

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Technology degree in Biotechnology
Interdisciplinary project

By

ANKIREDDY MARY MAMATHA (41230009)

NIMMAGADDA HEMA SRI (41230036)



**DEPARTMENT OF BIOTECHNOLOGY
SCHOOL OF BIO AND CHEMICAL ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with grade “A++” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF BIOTECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **Ankireddy Mary Mamatha** (Reg. No. 41230009), **Nimmagadda Hema Sri** (Reg. No. 41230036) who carried out the Interdisciplinary Project entitled "**AUTOMATED ANESTHESIA CONTROL SYSTEM AND PATIENT MONITORING**" under our supervision from January 2024 to April 2024.

Internal Guide

Dr. J. THEBORAL, PhD

Head Of the Department

Dr. V. RAMESH KUMAR, M.Phil., Ph.D.

Submitted for Viva voce Examination held on 06/05/24

6/5/24
Internal Examiner

External Examiner

DECLARATION

We, Ankireddy Mary Mamatha (Reg. No. 41230009) and Nimmagadda HemaSri (Reg. No 41230036) hereby declare that the Project Report entitled "AUTOMATED ANESTHESIA CONTROL SYSTEM AND PATIENT MONITORING" done by us under the guidance of Dr. J. THEBORAL, PhD is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Biotechnology.

DATE: 06/05/24

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my 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.

I convey my sincere thanks to **Dr.Y.Swarnalatha, M.Sc., Ph.D.**, Dean, Higher Studies and **Dr.V.Ramesh Kumar, M.Phil., Ph.D.**, Head of the Department, Department of Biotechnology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my internal Guide **Dr. THEBORAL,PhD.**, and external Guide **Dr.V.Ramesh Kumar,M.Phil.,Ph.D.**, for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I also express my thanks to all Teaching and Non-teaching staff members of the Department of Biotechnology who were helpful in many ways for the completion of the project work.

ABSTRACT

Providing painless surgery and delivering an accurate dose of anesthesia to the patient plays a very crucial role in any major surgeries. Failing to provide an accurate dose for the patient may show adverse effects and postoperative complications. In the case of major surgeries which could take a longer period, the complete dosage of anesthesia could not be administered in a single dose to the patient since it may show lethal complications. Administering less dose of anesthesia makes the patient regain consciousness during the surgery. It is not easy for the anesthetist to deliver an accurate dose of anesthesia at regular intervals of time. To overcome such complications during surgeries a computer-controlled syringe is designed to deliver an accurate dose of anesthesia at regular intervals of time with constant speed based on the control parameters like temperature, Heart rate and pulse rate. Therefore, this project aims to introduce an automatic anesthesia control system integrating with monitoring parameters using Arduino Uno and with the Blynk Software which is designed for IoT which can control the hardware remotely and display the sensor data. This instrument helps the anesthesiologist to control the anesthesia dosage to the patient and monitors the patient parameters, which is helpful to monitor the patient conditions and give the right dosage of the Anesthesia to the patient based on the heart rate and the pulse rate.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	
	LIST OF FIGURES	
	LIST OF TABLES	
	LIST OF ABBREVIATIONS	
1	INTRODUCTION	
	1.1 Overview	
	1.2 Objective of the Project	
2	LITERATURE SURVEY	
3	AIM AND SCOPE	
	3.1 Aim of the Project	
	3.2 Scope of the Project	
4	MATERIALS AND METHODS USED	
	4.1 Hardware Tools	
	4.1.1 NODEMCU (ESP-8266 12E)	
	4.1.1.1 Family of Boards (ESP-NN)	
	4.1.1.2 Features	
	4.1.1.3 AT Commands	
	4.1.1.4 AT Command Processor	
	4.1.1.5 ESP-12E Pin Design	
	4.1.1.6 Pin Descriptions	

	4.1.1.7 Memory Organization
	4.1.2 Pulse Sensor
	4.1.3 DHT11 Module
	4.1.3.1 Technical Specification
	4.1.3.2 Applications
	4.1.3.3 Typical Circuit
	4.1.3.4 Communication Description
	4.1.3.5 Applications Information
	4.1.4 Servo Motor
	4.1.4.1 Types of Servo Motor
	4.1.4.2 Working Principle of AC
	4.1.4.3 Servo Mechanism
	4.1.4.4 Working Principle of DC
	4.1.4.5 Controlling Servo Motor
	4.2 Software Tools
	4.2.1 Arduino IDE
	4.2.1.1 Arduino Installation
	4.2.2 BLYNK Software
	4.2.2.1 Features
	4.2.2.2 Characteristics of Blynk
	4.3 Methodology and Implementation
	4.3.1 Source Code
	4.3.2 Hardware Model

5 RESULTS

6 SUMMARY AND CONCLUSIONS

6.1 Summary

6.2 Conclusion

7

REFERENCES

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
4.1	NODEMCU (ESP8266-12E)	
4.2	ESP-12E Pin Design	
4.3	Pulse Sensor	
4.4	PPG Wave	
4.5	Pulse Sensor Waveform	
4.6	Pulse Sensor Kit	
4.7	op-amp Representation	
4.8	Arduino pin AO Representation	
4.9	Rectangular pulse wave form	
4.10	DHT11 Sensor	
4.11	Views of the Sensor	
4.12	Circuit diagram of DHT11	
4.13	Data Timing Diagram	
4.14	Starts the Signal	
4.15	Data receiving and start Signal	
4.16	Big Data Format Signal	
4.17	Servo Motor	
4.18	DC Motor	
4.19	Synchronous type AC Servo Motor	

4.20	Induction type AC Servo Motor
4.21	Working principle of AC Servo Motor
4.22	Working principle of DC Servo Motor
4.23	USB Cable
4.24	Block Diagram for Patient Monitoring
4.25	Block Diagram for Syringe Setup
4.26	Photograph of Hardware Model
5.1	Hardware Model
5.2	BLYNK Interface

LIST OF TABLES

TABLE NO	NAME OF THE TABLE	PAGE NO
4.1	Command and Description of NODEMCU	
4.2	Pin Description and Functions	
4.3	Pin Mode	
4.4	Dimension of ESP-12E	
4.5	Technical Specifications	

LIST OF SYMBOLS AND ABBEREVIATIONS

ABBREVIATIONS	FULL FORM
RX	Receiver
TX	Transmitter
VCC	Voltage common collector
GND	Ground
IOT	Internet of Things
NODEMCU	NODE Micro Controller Unit
Wi-Fi	Wireless Fidelity
DHT11	Digital Temperature and Humidity Sensor
HRV	Heart rate Variability
PTT	Pulse Transit Time

CHAPTER-1

INTRODUCTION

1.1 OVERVIEW

Anesthesia is a medical drug that can be given to the patient to avoid pain during surgeries. It is the greatest discovery in the medicine which provides comfort to both patient and doctor while performing surgeries. Anesthesia is divided into three types General, Regional and Local Anesthesia. General anesthesia is mainly used for all major surgeries to make the patient unconscious, Local can be given to the numb small area of the body whereas Regional can be given to block pain in an area of the body such as an arm or leg. Delivery of anesthesia can be given to the patients in two ways either through inhalation or intravenous. Giving intravenous anesthesia shows a quick response in patients. Anesthesiologists are overloaded with multiple tasks like to deliver anesthesia continuously to maintain unconsciousness till the end of surgery and also need to monitor physiological parameters side by side. This Computer based system uses a syringe pump to deliver the right amount of anesthesia to the patient. The anesthetist can set the desired amount of anesthesia that can be given to the patient based on the control parameters of the patient. Once the Arduino Uno receives the signal it activates the motor driver to drive the syringe pump at the preset intervals. The anesthesia is delivered to the patient according to the rotation of the servo motor. After administration of anesthesia, the vital parameters like Temperature, Heart rate, and Pulse are monitored side by side. If they are under the normal state then the second dose of anesthesia will be injected. On the onset of abnormality, the doctor will be notified and anesthesia delivery would continue only if everything is under normal. Additionally, these parameters are checked by corresponding sensors as well as with the Blynk software. This integration of monitoring parameters increases the patient's safety and keeps the anesthesiologists at ease.

1.2 OBJECTIVE OF THE PROJECT

The objective of implementing an automatic anesthesia control system with integrated patient monitoring is to revolutionize anesthesia delivery during surgical procedures by prioritizing patient safety, optimizing sedation levels, and streamlining workflow efficiency. This project aims to develop a sophisticated system that utilizes real-time monitoring data to automatically adjust anesthesia dosages based on individual patient responses and surgical requirements. By leveraging advanced monitoring technologies, such as electrocardiography, pulse oximetry, capnography, and blood pressure monitoring, the system aims to enhance the detection of physiological changes and promptly alert healthcare providers to potential complications. Through personalized anesthesia protocols and data-driven decision-making, this initiative seeks to improve patient outcomes, minimize risks associated with anesthesia administration, and foster technological innovation within perioperative medicine. Ultimately, the objective is to establish a comprehensive and efficient approach to anesthesia management that enhances overall patient care and safety in surgical settings.

In summary, the objectives of this project encompass a holistic approach to anesthesia management, emphasizing patient safety, workflow optimization, data-driven decision-making, and technological innovation. By achieving these objectives, the automatic anesthesia control system with patient monitoring aims to elevate standards of care in surgical settings, ultimately improving outcomes for patients and enhancing the practice of perioperative medicine.

CHAPTER-2

LITERATURE SURVEY

The literature survey conducted by Dr. R. Anitha and colleagues, including Ms. B. Nivetha, R. Abinaya, M. Dharshan Keba, and S.A. Sree Vishwa, on automated anesthesia control systems with patient monitoring provides a comprehensive overview of existing research and advancements in this field. Their survey encompasses studies that explore various aspects of automated anesthesia delivery, such as closed-loop systems, real-time monitoring technologies, adaptive control algorithms, and integration of advanced sensors for physiological data collection. The survey likely includes key publications that demonstrate the efficacy of automated systems in optimizing anesthesia dosing, enhancing patient safety, and improving surgical outcomes. By synthesizing findings from a range of studies, this literature survey aims to identify gaps, challenges, and opportunities for further development and implementation of automated anesthesia technologies in clinical practice. The work by Dr. Anitha and team contributes valuable insights that can inform the design, validation, and adoption of automated anesthesia control systems, ultimately advancing the field of perioperative medicine and enhancing patient care.

The study by Chazot et al. (2017) on closed-loop systems for anesthesia delivery serves as a foundational work in the field, emphasizing the automated titration of volatile anesthetics based on processed EEG signals and hemodynamic parameters. A comprehensive literature survey building upon this study would encompass a broad range of related research topics and advancements. This survey would explore various algorithmic approaches employed in closed-loop anesthesia systems, including adaptive control strategies and machine learning algorithms utilized to optimize anesthesia depth and maintain hemodynamic stability in real-time. Additionally, the survey would delve into the integration of advanced monitoring technologies beyond EEG and hemodynamics, such as spectral index (BIS), entropy measures, and respiratory variables, for comprehensive closed-loop anesthesia control. Clinical outcomes and safety profiles of closed-loop systems would be assessed through a review of clinical trials and observational studies, focusing on improvements in anesthesia depth control, reduced

drug consumption, and prevention of intraoperative awareness compared to manual techniques. Furthermore, the literature survey would investigate user acceptance, workflow integration, economic analyses, and regulatory considerations surrounding closed-loop anesthesia systems, aiming to provide a holistic understanding of the clinical utility, technical advancements, and implementation challenges associated with automated anesthesia delivery technologies. This survey seeks to inform evidence-based practice, drive innovation, and optimize perioperative care to enhance patient outcomes and safety across diverse healthcare settings.

The publication by Gong et al. (2021) represents a significant advancement in the application of machine learning algorithms for predicting intraoperative hemodynamic instability and guiding anesthesia management decisions. This study likely focused on leveraging real-time data analytics to identify patterns and early indicators of hemodynamic instability during surgery. By analyzing a combination of physiological parameters such as blood pressure, heart rate variability, and other relevant vital signs, Gong et al. demonstrated the feasibility of using machine learning models to predict potential hemodynamic fluctuations before they manifest clinically. One key aspect of this study would be the development and validation of machine learning algorithms tailored for intraoperative use. Gong et al. likely trained these algorithms on large datasets of intraoperative physiological data to enable accurate prediction of hemodynamic instability in real-time. The study likely evaluated the performance of these algorithms in terms of sensitivity, specificity, and predictive value, aiming to demonstrate their effectiveness in improving patient safety and surgical outcomes.

Overall, the publication by Gong et al. underscores the transformative impact of real-time predictive analytics and machine learning in anesthesia practice. This work contributes valuable insights into the future of data-driven anesthesia management, highlighting opportunities to leverage advanced technologies for proactive patient care and improved surgical outcomes.

CHAPTER-3

AIM AND SCOPE

3.1 AIM OF THE PROJECT

The aim of developing an automated anesthesia control system with patient monitoring is to revolutionize the process of anesthesia delivery during surgical interventions by prioritizing safety, precision, and efficiency. This project seeks to achieve several specific objectives aimed at enhancing perioperative care. Firstly, the system aims to optimize anesthesia administration through the implementation of a closed-loop mechanism that automatically adjusts drug dosages based on real-time patient monitoring data, ensuring precise sedation levels tailored to individual needs. Secondly, by integrating advanced monitoring technologies, such as continuous vital sign monitoring and physiological parameter assessment, the system aims to enhance patient safety by promptly detecting and responding to deviations from normal parameters, thus minimizing anesthesia-related complications. Moreover, this project aims to improve workflow efficiency by automating routine tasks associated with anesthesia management, allowing healthcare providers to focus more on patient care and coordination. Additionally, the system will enable personalized anesthesia protocols by leveraging data-driven algorithms to adapt sedation levels according to specific patient characteristics and surgical requirements, thereby optimizing comfort and safety. By collecting and analyzing comprehensive anesthesia and monitoring data, this initiative aims to contribute valuable insights for evidence-based medicine and drive further technological innovation in perioperative care. Ultimately, the goal of this project is to establish a sophisticated and reliable automated anesthesia control system that not only enhances patient outcomes and safety but also sets new standards for efficiency and innovation in surgical practice.

3.2 SCOPE OF THE PROJECT

The scope of the project for an automated anesthesia control system with patient monitoring is comprehensive and encompasses multiple facets of system design, development, implementation, and evaluation within the perioperative care context. Firstly, the project involves designing and developing the architecture of the automated

system, which includes selecting and integrating appropriate hardware components (e.g., monitoring devices, infusion pumps) and developing software components such as closed-loop control algorithms and user interfaces. Sensor integration is a critical aspect, focusing on capturing real-time physiological data including vital signs like heart rate, blood pressure, oxygen saturation, EEG, and respiratory parameters. Algorithm development and optimization are key areas, aiming to create responsive and adaptable algorithms for anesthesia delivery based on monitored parameters. The project also involves designing an intuitive user interface for healthcare providers to interact with the system, enabling real-time monitoring and adjustment of anesthesia parameters. Rigorous testing and validation will be conducted to assess system performance, reliability, and safety in simulated and clinical environments. Clinical integration and evaluation will follow, including trials or studies to assess patient outcomes, anesthesia quality, and user satisfaction. Regulatory compliance and standards adherence are critical, ensuring the system meets necessary approvals for medical device use. Finally, the project includes ongoing training, education, and continuous improvement efforts based on user feedback and technological advancements, aiming to continually enhance the effectiveness, safety, and usability of the automated anesthesia control system. This comprehensive scope aims to develop a sophisticated and reliable system that optimizes anesthesia delivery, enhances patient safety, and contributes to advancements in perioperative medicine.

CHAPTER-4

MATERIALS AND METHODS USED

4.1 HARDWARE TOOLS

We divided our task into two parts to make the task easy and simple and to avoid complexity and make it error free. The first is the hardware parts which includes following components:

- NODEMCU (ESP-8266 12E) Wi-Fi Module
- Pulse Sensor
- Temperature and Humidity DHT11
- Servo Motor

The second is the software parts which comprises of the following main components:

- Arduino IDE
- Blynk Software

4.1.1 NODEMCU (ESP-8266 12E) Wi-Fi Module

ESP-8266 12E Wi-Fi module is developed by Ai-thinker Team. core processor ESP8266 in smaller sizes of the module encapsulates Tensilica L106 integrates industry-leading ultra-low power 32-bit MCU micro, with the 16-bit short mode, Clock speed support 80 MHz, 160 MHz, supports the RTOS, integrated Wi-Fi MAC/BB/RF/PA/LNA, on-board antenna. The module supports standard IEEE802.11 b/g/n agreement, complete TCP/IP protocol stack. Users can use the add modules to an existing device networking, or building a separate network controller. ESP8266 is high integration wireless SOCs, designed for space and power constrained mobile platform designers. It provides unsurpassed ability to embed Wi-Fi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement.

ESP8266EX offers a complete and self-contained Wi-Fi networking solution; it can be used to host the application or to offload Wi-Fi networking functions from another application processor. When ESP8266EX hosts the application, it boots up directly from an external flash.

ESP8266EX is among the most integrated Wi-Fi chip in the industry; it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.

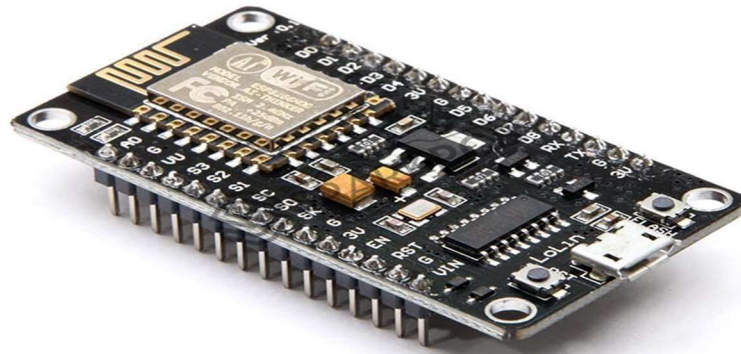


Figure 4.1 NodeMcu (ESP8266-12E)

The ESP8266 has seen a wide adoption as a cost-effective solution for IOT and Wi-Fi-capable devices. The ESP8266 was developed by Shanghai-based Espressif systems, as a Serial (UART) to Wi-Fi SoC (System on a Chip) based around a Tensilica Xtensa LX3DPU. This tiny IC includes an RF front end, RAM, and (usually) an onboard TCP/IP stack that allows it ready to connect to a nearby Access Point, to act as an Access Point itself, or both.

4.1.1.1 Family of Breakout Boards (ESP-NN):

Quickly after launch, a variety of breakout boards for the ESP8266 started becoming available. The most popular ones have been the ESP-NN series, which typically integrate the SOC along with Flash RAM, a crystal, and even an onboard antenna. The most salient distinction between different ESP-NN models are the pins that are broken out from the ESP8266

As the ESP8266 was developed as a Serial to Wi-Fi adapter, its firmware implemented an interpreter for AT commands. Thus, initial usage of the IC was limited to using either a USB to Serial adapter, or a separate microcontroller (e.g., ATmega328) to issue AT commands over the ESP8266's Serial UART interface. For this reason, the ESP-01 board quickly became popular amongst the ESP8266 community because of its 2×4, 0.1in-pitch connector that can be easily wired to a USB to Serial adapter.

The connector gave access to the pins used for serial communication, namely RX and TX, as well as 4 control pins, GPIO0, GPIO2, CH_PD and RST (reset), along with VCC and GND.

However, other ESP-NN boards offer access to a wider variety of pins, although their packaging is of a custom Surface-Mount Device, with castellated pins as seen in the documentation page linked above. From the different ESP-NN boards, we began to experiment with the ESP-12E. The main reason is that this module was the one chosen by the developers of the Node MCU project for their hardware Dev Kit 1.0

Because the ESP8266 provides a cost-effective solution to the rapidly growing market of internet-connected projects and devices (i.e., the so-called Internet of Things), it has become one of the most popular development platforms over the past year and a half. In consequence, a dedicated community has formed around the platform which has been focused on improving its functionality.

For starters, different firmware options have been ported to run on the ESP8266, effectively taking it from a simple Serial to Wi-Fi adapter into a fully functional microcontroller with access to its GPIO and hardware-based functions like PWM, I2C, 1-Wire communication, and ADC; all this, of course, in addition to maintaining its Wi-Fi capabilities.

4.1.1.2 Features:

- 802.11 b/g/n
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC

- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Wi-Fi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and IOS devices
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation and 0.4s guard interval Shenzhen Anxinke Technology
- Deep sleep power < 5uA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- +20dBm output power in 802.11b mode • Operating temperature range -40C ~ 125C

Applications:

- Smart power plugs
- Home automation
- Mesh network
- Industrial wireless control
- Baby monitors
- IP Cameras
- Sensor networks
- Wi-Fi location-aware devices
- Security ID tags
- Wi-Fi position system beacons.

4.1.1.3 AT Commands:

Table 4.1 Command and Description of NODEMCU ESP8266

Command	Description
AT	Test AT start up
AT+RST	Restart module
AT+GMR	View version Info
AT+GSLP	Enter deep sleep mode
ATE	AT commands echo or not
AT+RESTORE	Factory Reset
AT+UART	UART Configuration
AT+UART_CUR	UART current configuration
AT+UART_DEF	UART default configuration, save to flash
AT+SLEEP	Sleep mode
AT+RFPOWER	Set maximum value of RF TX power
AT+RFVVD	RF TX power according to VDD33

A few different firmware options are available for the ESP8266. These allow us to access the module in different ways, as you can see below.

4.1.1.4 AT Command Processor (Default):

The quickest way to get started with the ESP8266 is to use its original firmware, which allows it to process any AT commands that it receives over its Serial UART interface. The biggest advantage of this option is that we need not be familiar with any specific language or framework to use the module. We can simply send it a series of commands to achieve our goal. The downside to this is that we need either an additional microcontroller involved or a USB to Serial adapter to send the necessary commands.

Whereas the AT commands are the standard way of communicating with wireless-capable ICs (e.g., Bluetooth, Wi-Fi, GSM), they pose the limitation of needing another module to run the application that specifies these commands accordingly. However, if we could run the application within the ESP8266 itself then we'd have everything self-contained by a single IC. Fortunately, Espressif made a Software Development Kit (SDK) available that allowed users to flash different firmware options.

4.1.1.5 ESP-12E Pin design:

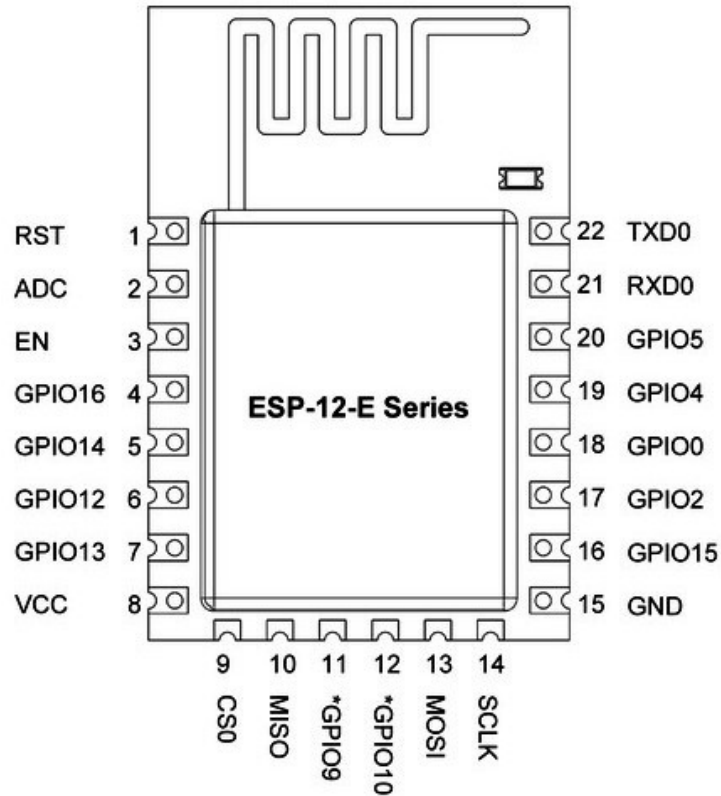


Figure 4.2: ESP-12E Pin design

4.1.1.6 Pin Descriptions:

Table 4.2: Pin Descriptions and Functions

NO.	Pin Name	Function
1	RST	Reset the module
2	ADC	A/D Conversion result. Input voltage range 0-1v,scope:0-1024
3	EN	Chip enable pin. Active high
4	IO16	GPIO16; can be used to wake up the chipset from deep sleep mode.
5	IO14	GPIO14; HSPI_CLK
6	IO12	GPIO12; HSPI_MISO
7	IO13	GPIO13; HSPI_MOSI; UART0_CTS
8	VCC	3.3V power supply (VDD)
9	CS0	Chip selection
10	MISO	Salve output Main input

11	IO9	GPIO9
12	IO10	GPIO10
13	MOSI	Main output slave input
14	SCLK	Clock
15	GND	GND
16	IO15	GPIO15; MTDO; HSPICS; UART0_RTS
17	IO2	GPIO2; UART1_TXD
18	IO0	GPIO0
19	IO4	GPIO4
20	IO5	GPIO5
21	RXD	UART0_RXD; GPIO3
22	TXD	UART0_TXD; GPIO1

Table 4.3 Pin Mode

Mode	GPIO15	GPIO0	GPIO2
UART	Low	Low	High
Flash Boot	Low	High	High

Table 4.4: Dimension of ESP-12E Wi-Fi Module

Length	Width	Height	PAD Size (Bottom)	Pin Pitch
16 mm	24mm	3 mm	0.9 mm x 1.7 mm	2mm

ESP8266EX is embedded with Tensilica L106 32-bit micro controller (MCU), which features extra low power consumption and 16-bit RSIC. The CPU clock speed is 80MHz. It can also reach a maximum value of 160MHz. ESP8266EX is often integrated with external sensors and other specific devices through its GPIOs; codes for such applications are provided in examples in the SDK.

4.1.1.7 Memory Organization:

Internal SRAM and ROM:

ESP8266EX Wi-Fi SOC is embedded with memory controller, including SRAM and ROM. MCU can visit the memory units through iBus, dBus, and AHB interfaces. All memory units can be visited upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

According to our current version of SDK provided, SRAM space that is available to users is assigned as below:

- RAM size < 36kB, that is to say, when ESP8266EX is working under the station mode and is connected to the router, programmable space accessible to user in heap and data section is around 36kB.)
- There is no programmable ROM in the SOC, therefore, user program must be stored in an external SPI flash.

External SPI Flash:

This module is mounted with an 4 MB external SPI flash to store user programs. If larger definable storage space is required, a SPI flash with larger memory size is preferred. Theoretically speaking, up to 16 MB memory capacity can be supported.

4.1.2 PULSE SENSOR

Pulse Sensor is a well-designed plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart rate data into their projects. The sensor clips onto a fingertip or earlobe and plugs right into Arduino. It also includes an open-source monitoring app that graphs your pulse in real time.

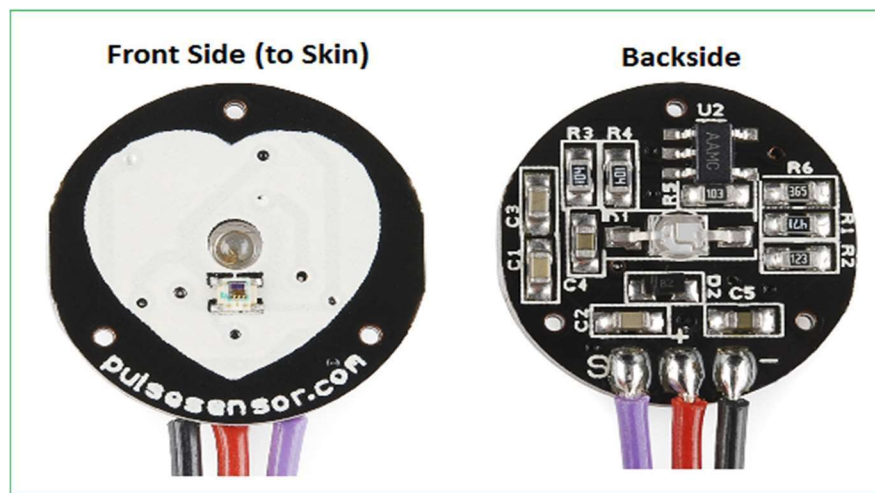


Figure 4.3: Pulse Sensor

The Pulse Sensor that we make is essentially a photoplethysmography, which is a well-known medical device used for non-invasive heart rate monitoring. Sometimes, photoplethysmography measure blood-oxygen levels (SpO₂), sometimes they don't. The heart pulse signal that comes out of a photoplethysmography is an analog fluctuation in voltage, and it has a predictable wave shape as shown in figure. The depiction of the pulse wave is called a photoplethysmogram, or PPG. Our latest hardware version, Pulse Sensor Amped, amplifies the raw signal of the previous Pulse Sensor, and normalizes the pulse wave around V/2 (midpoint in voltage).

Pulse Sensor Amped responds to relative changes in light intensity. If the amount of light incident on the sensor remains constant, the signal value will remain at (or close to) 512 (midpoint of ADC range). More light and the signal go up. Less light, the opposite. Light from the green LED that is reflected back to the sensor changes during each pulse.

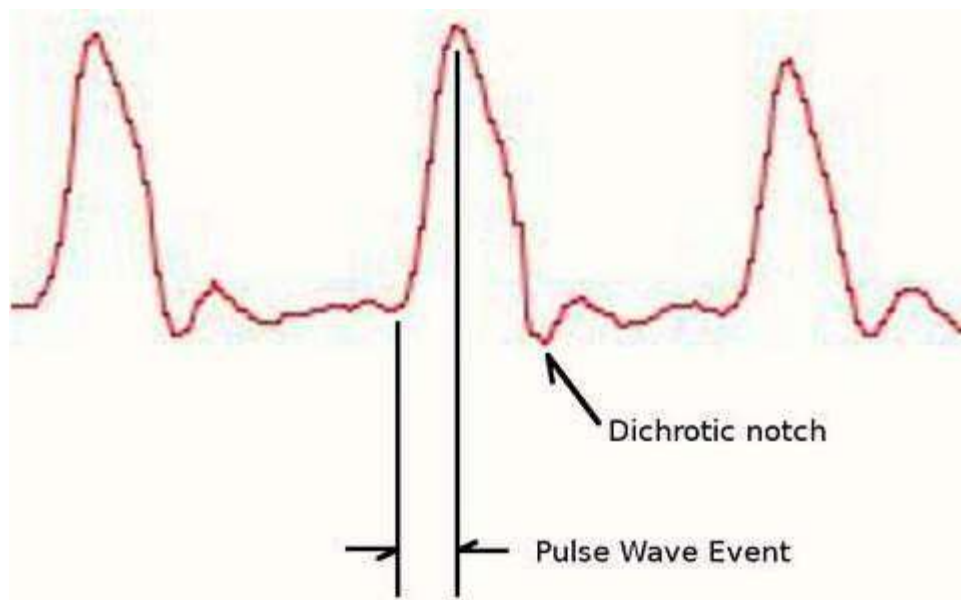


Figure 4.4 PPG Wave

Our goal is to find successive moments of instantaneous heart beat and measure the time between, called the Inter Beat Interval (IBI). By following the predictable shape and pattern of the PPG wave, we are able to do just that.

Now, we're not heart researchers, but we play them on this blog. We're basing this page on Other People's Research that seem reasonable to us (references below). When the

heart pumps blood through the body, with every beat there is a pulse wave (kind of like a shock wave) that travels along all arteries to the very extremities of capillary tissue where the Pulse Sensor is attached. Actual blood circulates in the body much slower than the pulse wave travels. Let's follow events as they progress from point 'T' on the PPG below. A rapid upward rise in signal value occurs as the pulse wave passes under the sensor, then the signal falls back down toward the normal point. Sometimes, the dichroic notch (downward spike) is more pronounced than others, but generally the signal settles down to background noise before the next pulse wave washes through. Since the wave is repeating and predictable, we could choose almost any recognizable feature as a reference point, say the peak, and measure the heart rate by doing math on the time between each peak.

This, however, can run into false readings from the dichroic notch, if present, and may be susceptible to inaccuracy from baseline noise as well. There are other good reasons not to base the beat-finding algorithm on arbitrary wave phenomena. Ideally, we want to find the instantaneous moment of the heartbeat. This is important for accurate BPM calculation, Heart Rate Variability (HRV) studies, and Pulse Transit Time (PTT) measurement. And it is a worthy challenge! People Smarter than us (note1) argue that the instantaneous moment of heart beat happens at some point during that fast upward rise in the PPG waveform.

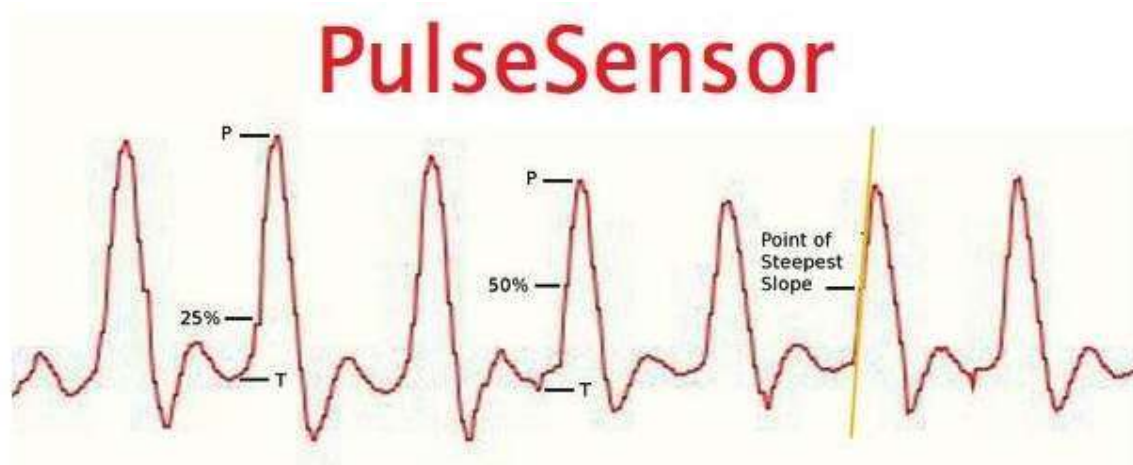


Figure 4.5 Pulse Sensor Waveform

Some heart researchers say it's when the signal gets to 25% of the amplitude, some say when it's 50% of the amplitude, and some say it's the point when the slope is steepest during the upward rise event. This version 1.1 of Pulse Sensor code is designed to measure the IBI by timing between moments when the signal crosses 50% of the wave amplitude during that fast upward rise. The BPM is derived every beat from an average of the previous 10 IBI times. Here's how we do it!

First off, it's important to have a regular sample rate with high enough resolution to get reliable measurement of the timing between each beat. To do this, we set up Timer2, an 8-bit hardware timer on the ATmega328 (UNO), so that it throws an interrupt every other millisecond. That gives us a sample rate of 500Hz and beat-to-beat timing resolution of 2mS (note2). This will disable PWM output on pin 3 and 11. Also, it will disable the tone() command. This code works with Arduino UNO or Arduino PRO or Arduino Pro Mini 5V or any Arduino running with an ATmega328 and 16MHz clock.

The Pulse Sensor Kit includes:

- 1) A 24-inch Color-Coded Cable, with (male) header connectors. You'll find this makes it easy to embed the sensor into your project, and connect to an Arduino. No soldering is required.
- 2) An Ear Clip, perfectly sized to the sensor. We searched many places to find just the right clip. It can be hot glued to the back of the sensor and easily worn on the earlobe.
- 3) 2 Velcro Dots. These are 'hook' side and are also perfectly sized to the sensor. You'll find these Velcro dots very useful if you want to make a Velcro (or fabric) strap to wrap around a fingertip. Velcro strap to wrap the Pulse Sensor around your finger.
- 4) 3 Transparent Stickers. These are used on the front of the Pulse Sensor to protect it from oily fingers and sweaty earlobes.
- 5) The Pulse Sensor has 3 holes around the outside edge which make it easy to sew it into almost anything.

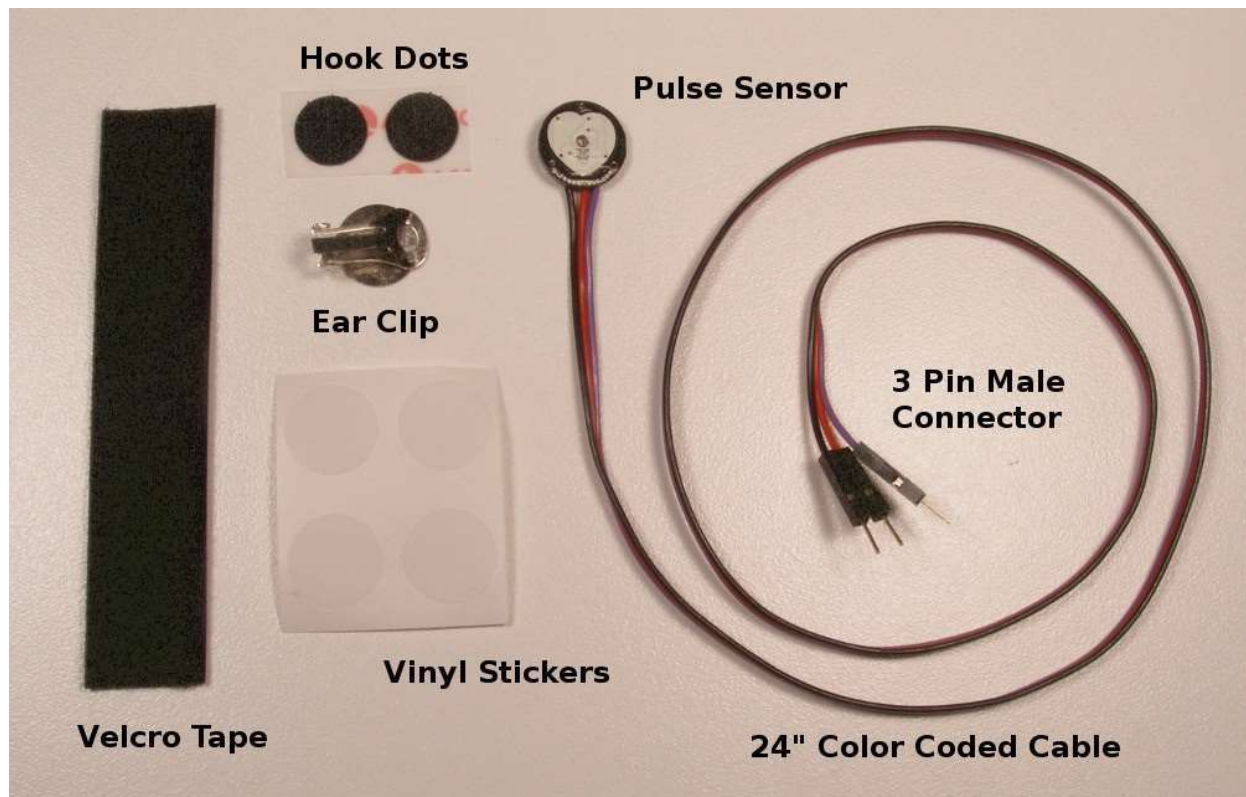


Figure 4.6 Pulse Sensor Kit

Taking your pulse is as simple as holding a finger to your neck or wrist and timing the beats with your watch. But if you want to record the data or use it to trigger events, you need to turn that mechanical pulsing action into an electrical signal. This pulse sensor fits over a fingertip and uses the amount of infrared light reflected by the blood circulating inside to do just that.

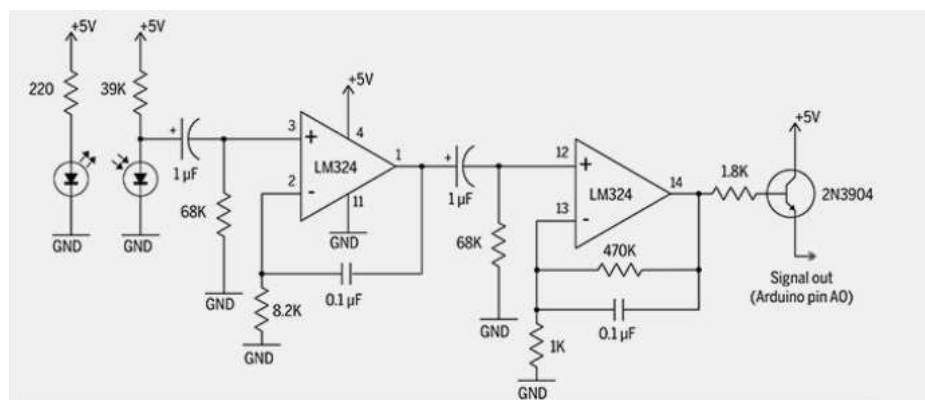


Figure 4.7 Schematic with symbolic op-amp representation

The sensor itself consists of an infrared emitter and detector mounted side-by-side and pressed closely against the skin. When the heart pumps, blood pressure rises sharply, and so does the amount of infrared light from the emitter that gets reflected back to the detector. The detector passes more current when it receives more light, which in turn causes a voltage drop to enter the amplifier circuitry. This design uses two consecutive operational amplifiers (“op-amps”) to establish a steady baseline for the signal, emphasize the peaks, and filter out noise. Both op-amps are contained in a single integrated circuit (IC or “chip”), and hooking them up is really just a matter of interconnecting the pins correctly.

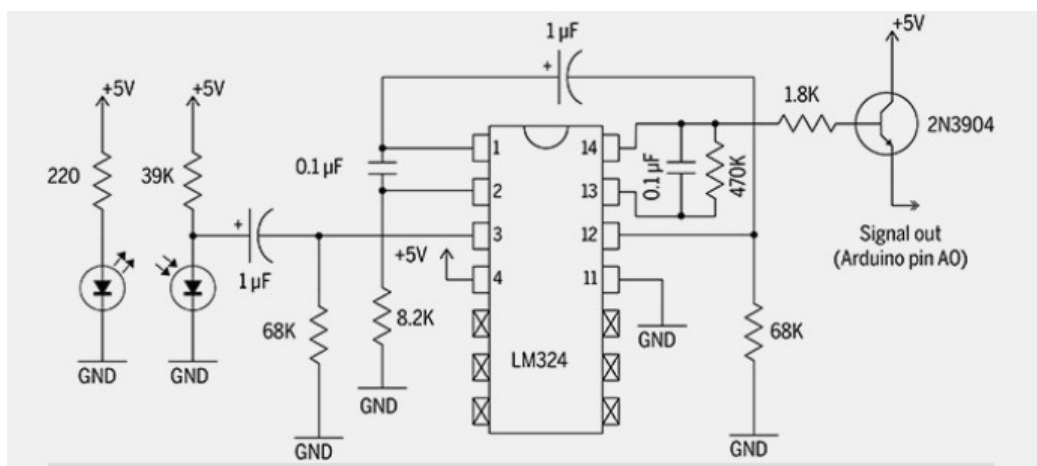


Figure 4.8: Arduino Pin AO Representation

The two op-amps output a clean but weak signal which is amplified by the transistor before output. The complete pulse sensor is a three-wire device that runs on 5V and outputs signal on the white wire. You can visualize and/or record this signal in a number of ways, but we’ve chosen to connect to a personal computer through Arduino, mostly because of the ease of integrating Processing, which in turn is very handy for visualization. But you don’t really need an Arduino to use the sensor. More on that below.

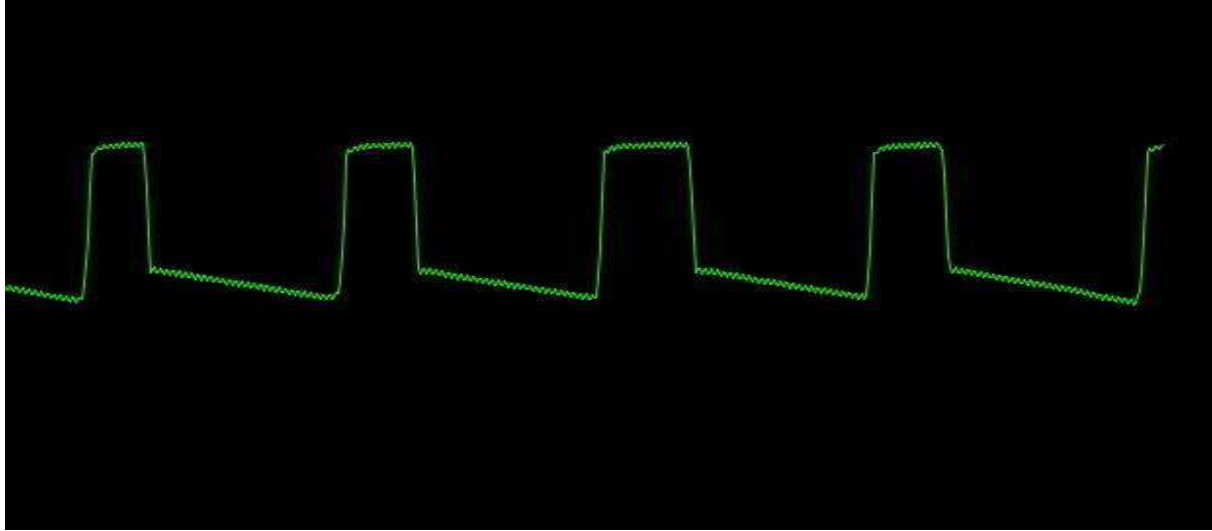


Figure4.9: Rectangular Pulse Waveform

4.1.3 Temperature and Humidity Module DHT11

DHT11 digital temperature and humidity sensor is a calibrated digital signal output of the temperature and humidity combined sensor. It uses dedicated digital modules capture technology and the temperature and humidity sensor technology to ensure that products with high reliability and excellent long-term stability. Sensor includes a resistive element and a sense of wet NTC temperature measurement devices and with a high-performance 8-bit microcontroller connected.

DHT11 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability. Its sensing elements are connected with 8-bit single-chip computer. Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in OTP memory. Small size & low consumption & long transmission distance (20m) enables DHT11 to be suited in all kinds of harsh application occasions. Single-row packaged with four pins, making the connection very convenient.

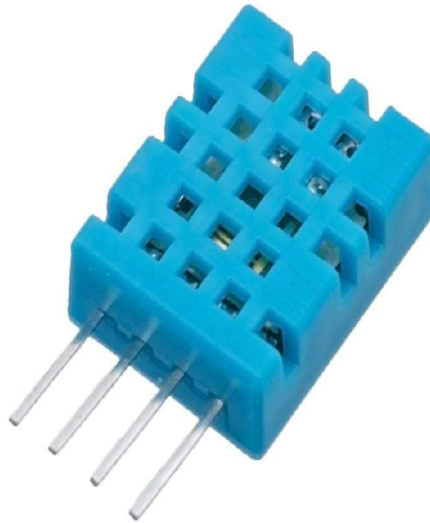


Figure 4.10:Arduino DHT11 Sensor

4.1.3.1 Technical Specification:

Table 4.5 Technical Specifications of this Model

MODEL	DHT11
Power supply	3-5.5V DC
Output signal	digital signal via single-bus
Sensing element	Polymer resistor
Measuring range	humidity 20-90%RH; temperature 0-50 Celsius
Accuracy	humidity $\pm 4\%$ RH (Max $\pm 5\%$ RH); temperature ± 2.0 Celsius
Resolution or sensitivity	humidity 1%RH; temperature 0.1Celsius
Repeatability	humidity $\pm 1\%$ RH; temperature ± 1 Celsius
Humidity hysteresis	$\pm 1\%$ RH
Long-term Stability	$\pm 0.5\%$ RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	size 12*15.5*5.5mm

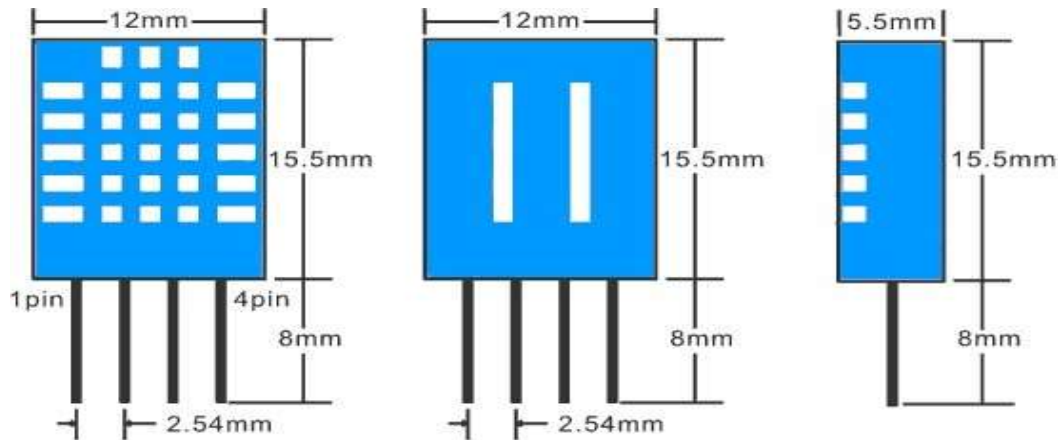


Figure 4.11 : Front View, Back View and Side view of the sensor

4.1.3.2 Applications:

HVAC, dehumidifiers, testing and inspection equipment, consumer goods, automotive, automation, data loggers, weather stations, home appliances, humidity regulator, medical and other relevant humidity measurement and control.

4.1.3.3 Typical circuit:

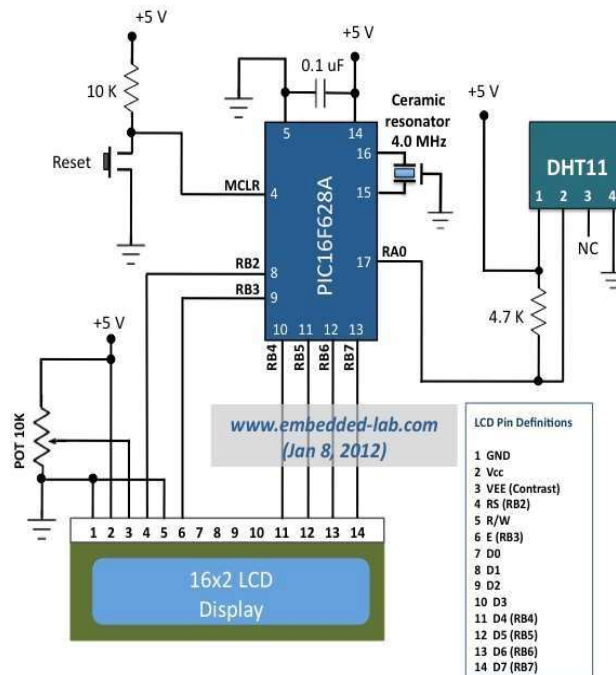


Figure 4.12 Typical Circuit Diagram of DHT11

Connecting the typical application circuit shown above the microprocessor and DHT11, DATA pull-up and microprocessor I/O port. 1. A typical application circuit recommended cable length shorter than 20 meters with a 5.1K pull-up resistor when greater than 20 meters when the pull-up resistor to reduce the actual situation. 2. When using a 3.3V voltage supply cable length must not be greater than 100cm. Otherwise it will lead to lack of line drop sensor supply, causing measurement bias. 3. Temperature and humidity values are read out every last measurement result, want to get real-time data, to be read twice in a row, but not recommended repeatedly read sensors, each sensor reading interval of more than 5 seconds to obtain accurate data.

4.1.3.4 Serial Communications Description (single-wire bidirectional):

Single Bus Description:

DHT11 device uses a simplified single-bus communication. Single bus that only one data line, the data exchange system, are controlled by a single bus is complete. Device (master or slave) through an open-drain or tri-state port is connected to the data line to allow the device to send data when not able to release the bus, and let other devices use the bus; single bus usually requires an external approximately 5.1k Ω pull up resistor, so that when the bus is idle, the state is high. Because they are master-slave structure, only the host calls a slave, a slave to answer, so the host access devices must strictly follow the sequence of a single bus, if there is a sequence of confusion, the device will not respond to the host

Single bus transfer data bit definition:

DATA is used for communication between the microprocessor and DHT11 and synchronization, single-bus data format, a 40-bit data transfer, high first-out. Data formats: 8bit humidity integer data + 8bit decimal data +8-bit temperature and humidity data + 8bit temperature decimal integer data +8-bit parity bit.

Parity bit data definition:

"8bit humidity decimal integer data + 8bit humidity temperature data +8-bit decimal integer data + 8bit temperature data" 8bit parity bit is equal to the result of the end of eight.

Example One : 40 receives the data to:

<u>0011 0101</u>	<u>0000 0000</u>	<u>0001 1000</u>	<u>0000 0000</u>	<u>0100</u>
<u>1101</u>				

High humidity 8	Low humidity 8	High temperature 8	Low temperature 8	Parity bit
-----------------	----------------	--------------------	-------------------	------------

Calculated as follows:

$0011\ 0101 + 0000\ 0000 + 0001\ 1000 + 0000\ 0000 = 0100\ 1101$

Receive data is correct:

Humidity: $0011\ 0101 = 35H = 53\%RH$

Temperature: $0001\ 1000 = 18H = 24^{\circ}C$

Example Two: The received data is 40:

<u>0011 0101</u>	<u>0000 0000</u>	<u>0001 1000</u>	<u>0000 0000</u>
<u>0100 1001</u>			

High humidity 8	High humidity 8	High temperature 8	High temperature 8
Parity bit			

Calculated as follows:

$0011\ 0101 + 0000\ 0000 + 0001\ 1000 + 0000\ 0000 = 0100\ 1101\ 01001001$ is not equal to 01001101

The received data is not correct, give up, again receiving data.

Hosts (MCU) after sending a start signal, DHT11 transition from a low-power mode to high-speed mode, the host until after the end of the start signal, DHT11 send a response signal, send 40bit data acquisition and trigger a letter. Signal transmission shown in fig.



The host reads temperature and humidity data from DHT11 always previous measurements, such as the two measured time interval is long, please read twice in a row is the second time in real time temperature and humidity values.

Communication between master and slave can be completed by the following steps (peripherals (such as a microprocessor) to read step DHT11 data). Step one: DHT11 after power (power after DHT11 1S to wait to cross the unstable state during this period can't send any commands), test environment temperature and humidity data, and record data while the data lines DATA DHT11 pulled by a pull-up resistor remains high; DHT11 this time the DATA pin is the input state, always detect external signals. Step two: Microprocessor I / O output while the output is set to low, and low retention time can't be less than 18ms, then the microprocessor I / O is set to enter the state, due to the pull-up resistor, the microprocessor I / O that the data lines DHT11 also will go high, waiting to answer DHT11 signals transmitted signal as shown:

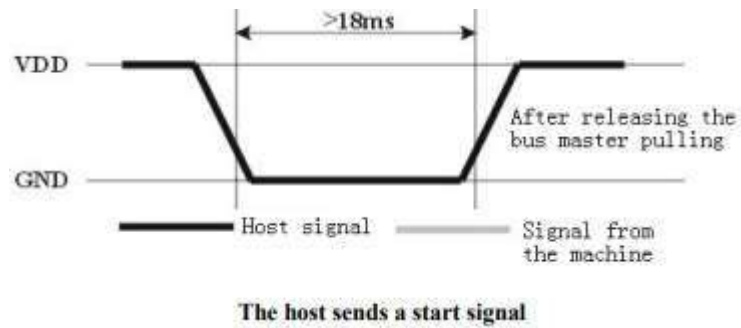


Figure 4.14: Starts the Signal

Step three:

DHT11 the DATA pin when external signals detected low, waiting for the external signal low end, after a delay DHT11 the DATA pin is an output, the output low as 80 microseconds response signal, followed by the output of 80 micro-notify the second high peripheral is ready to receive data, the microprocessor I / O at this time in the input state detecting I / O with low (DHT11 echo signal) to the wait for 80 microseconds high data receiving and sending signals as shown:

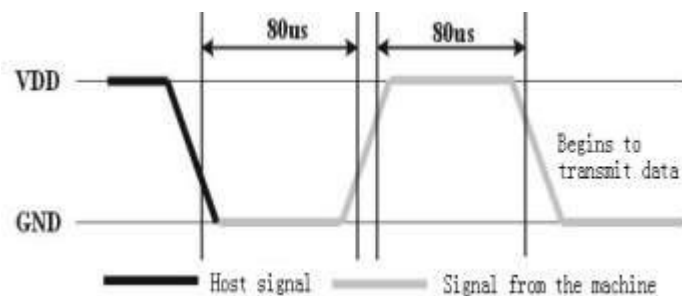


Figure 4.15: Data Receiving and start signal

Step four:

The 40 bit data output by the DHT11 DATA pin, the microprocessor according to the change of I/O level receive 40 bits of data, a data format of "0": high level and low level of 50 microseconds and 26-28 microsecond, format data "1": low level 50 microsecond plus 70 microsecond high. Bit data "0", "1" format signal as shown in fig:

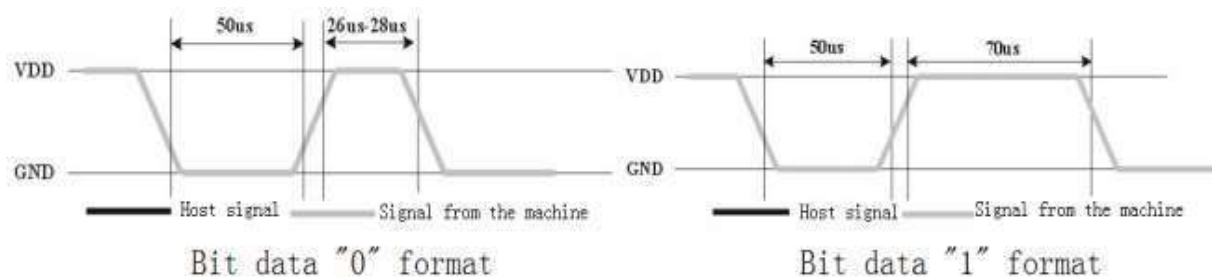


Figure 4.16: Bit data "0", "1" Format Signal

End signal:

DHT11 the DATA pin output 40-bit data, the continued output low 50 microseconds after the entry into the state, due to the pull-up resistor attendant goes high. But DHT11 temperature and humidity inside the test-retest data, and record the data, awaiting the arrival of an external signal.

4.1.3.5 Application Information:

Working and storage conditions:

The proposed scope of work may result in up to 3% RH temporary drift of the signal. Return to normal working conditions, the sensor calibration status will slowly recover. To speed up the recovery process can be found in "recovery process." The use of the product will accelerate the aging process for a long time under abnormal operating conditions. Avoid placing components on a long-term condensation and dry conditions and the following environments.

- A. smoke
- B. Acid or oxidizing gases such as sulfur dioxide, hydrochloric acid Recommended Storage Environment Temperature : 10~40°C Humidity : 60% RH or less

Effects of exposure to chemical substances:

Sensing resistive humidity sensor will be disturbed chemical vapor layer, the diffusion layer in the induction of chemicals may cause drift and measurement sensitivity. In a clean environment, slowly release contaminants out. The recovery process described below to accelerate the process. High concentrations of chemical pollution can cause damage to the sensor sensing layer completely.

Temperature Effect:

Relative humidity of the gas is largely dependent on temperature. Therefore, when measuring the humidity should be possible to ensure that the humidity sensor works at the same temperature. If you share a printed circuit board with electronic components heat released in the sensor should be installed as far as possible away from the electronic components, and installed at the bottom of the heat source, while maintaining a well-ventilated enclosure. To reduce the thermal conductivity sensor and a copper plating layer of the printed circuit board should be as minimal other portions, and leaving a gap between them.

Light effects:

Prolonged exposure to sunlight or strong ultraviolet radiation will reduce performance.

Recovery process:

Placed under extreme operating conditions or chemical vapor sensors, through the following process, you can return it to the state calibration. 45 °C and humidity under 70% RH conditions were maintained for more than 5 hours.

Wiring Precautions:

DATA signal wire quality will affect the communication distance and communication quality, we recommend using a high-quality shielded cable.

Soldering Information:

Manual welding, at a temperature of 300 °C maximum contact time must be less than 10 seconds.

4.1.4 SERVO MOTOR

A servo motor is an electrical device which can push or rotate an object with great precision. If you want to rotate an object at some specific angles or distance, then you use servo motor. It is just made up of simple motor which runs through servo mechanism. If motor is used is DC powered then it is called DC servo motor, and if it is AC powered motor then it is called AC servo motor. We can get a very high torque servo motor in a small and light weight package. Due to these features they are being used in many applications like toy car, RC helicopters and planes, Robotics, Machine etc.

Servo motors are rated in kg/cm (kilogram per centimeter) most hobby servo motors are rated at 3kg/cm or 6kg/cm or 12kg/cm. This kg/cm tells you how much weight your servo motor can lift at a particular distance. For example: A 6kg/cm Servo motor should be able to lift 6kg if the load is suspended 1cm away from the motor's shaft, the greater the distance the lesser the weight carrying capacity.

The position of a servo motor is decided by electrical pulse and its circuitry is placed beside the motor.



Figure 4.17 Servo Motor

A servo motor is a linear or rotary actuator that provides fast precision position control for closed-loop position control applications. Unlike large industrial motors, a servo motor is not used for continuous energy conversion. Servo motors have a high-speed response due to low inertia and are designed with small diameter and long rotor length. Then how do servo motors work? Servo motors work on servo mechanism that uses position feedback to control the speed and final position of the motor. Internally, a servo motor combines a motor, feedback circuit, controller and other electronic circuit. It uses encoder or speed sensor to provide speed feedback and position. This feedback signal is compared with input command position (desired position of the motor corresponding to a load), and produces the error signal (if there exist a difference between them). The error signal available at the output of error detector is not enough to drive the motor. So, the error detector followed by a servo amplifier raises the voltage and power level of the error signal and then turns the shaft of the motor to desired position.

4.1.4.1 Types of Servo Motors:

Basically, servo motors are classified into AC and DC servo motors depending upon the nature of supply used for its operation. Brushed permanent magnet DC servo motors are used for simple applications owing to their cost, efficiency and simplicity. These are best suited for smaller applications. With the advancement of microprocessor and power transistor, AC servo motors are used more often due to their high accuracy control.

DC Servo Motors:

A DC servo motor consists of a small DC motor, feedback potentiometer, gearbox, motor drive electronic circuit and electronic feedback control loop. It is more or less similar to the normal DC motor. The stator of the motor consists of a cylindrical frame and the magnet is attached to the inside of the frame.



Fig 4.18: DC Motor

The rotor consists of brush and shaft. A commutator and a rotor metal supporting frame are attached to the outside of the shaft and the armature winding is coiled in the rotor metal supporting frame. A brush is built with an armature coil that supplies the current to the commutator. At the back of the shaft, a detector is built into the rotor in order to detect the rotation speed. With this construction, it is simple to design a controller using simple circuitry because the torque is proportional to the amount of current flow through the armature. And also the instantaneous polarity of the control voltage decides the direction of torque developed by the motor. Types of DC servo motors include series motors, shunt control motor, split series motor, and permanent magnet shunt motor.

AC Servo Motors:

AC servo motors are basically two-phase squirrel cage induction motors and are used for low power applications. Nowadays, three phase squirrel cage induction motors have been modified such that they can be used in high power servo systems. The main difference between a standard split-phase induction motor and AC motor is that the squirrel cage rotor of a servo motor has made with thinner conducting bars, so that the motor resistance is higher. Based on the construction there are two distinct types of AC servo motors, they are synchronous type AC servo motor and induction type AC servo motor.

Synchronous-type AC servo motor:

Consists of stator and rotor. The stator consists of a cylindrical frame and stator core. The armature coil wound around the stator core and the coil end is connected to with a lead wire through which current is provided to the motor. The rotor consists of a permanent magnet and hence they do not rely on AC induction type rotor that has current induced into it. And hence these are also called as brushless servo motors because of structural characteristics.



Figure 4.19 : Synchronous-type AC servo motor

When the stator field is excited, the rotor follows the rotating magnetic field of the stator at the synchronous speed. If the stator field stops, the rotor also stops. With this permanent magnet rotor, no rotor current is needed and hence less heat is produced.

Also, these motors have high efficiency due to the absence of rotor current. In order to know the position of rotor with respect to stator, an encoder is placed on the rotor and it acts as feedback to the motor controller.

The induction-type AC servo motor structure is identical with that of general motor. In this motor, stator consists of stator core, armature winding and lead wire, while rotor consists of shaft and the rotor core that built with a conductor as similar to squirrel cage rotor.



Figure 4.20: Induction type AC Servo Motor

The working principle of this servo motor is similar to the normal induction motor. Again the controller must know the exact position of the rotor using encoder for precise speed and position control.

4.1.4.2 Working Principle of AC Servo Motor:

The schematic diagram of servo system for AC two-phase induction motor is shown in the figure below. In this, the reference input at which the motor shaft has to maintain at a certain position is given to the rotor of synchro generator as mechanical input θ . This rotor is connected to the electrical input at rated voltage at a fixed frequency.

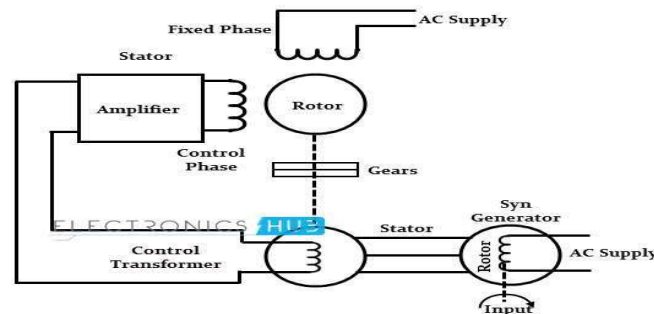


Figure 4.21: Working Principle of AC Servo Motor

The three stator terminals of a synchro generator are connected correspondingly to the terminals of control transformer. The angular position of the two-phase motor is transmitted to the rotor of control transformer through gear train arrangement and it represents the control condition alpha. Initially, there exist a difference between the synchro generator shaft position and control transformer shaft position. This error is reflected as the voltage across the control transformer. This error voltage is applied to the servo amplifier and then to the control phase of the motor. With the control voltage, the rotor of the motor rotates in required direction till the error becomes zero. This is how the desired shaft position is ensured in AC servo motors. Alternatively, modern AC servo drives are embedded controllers like PLCs, microprocessors and microcontrollers to achieve variable frequency and variable voltage in order to drive the motor. Mostly, pulse width modulation and Proportional-Integral-Derivative (PID) techniques are used to control the desired frequency and voltage. The block diagram of AC servo motor system using programmable logic controllers, position and servo controllers is given below.

4.1.4.3 Servo Mechanism:

It consists of three parts:

1. Controlled device
2. Output sensor
3. Feedback system

It is a closed loop system where it uses positive feedback system to control motion and final position of the shaft. Here the device is controlled by a feedback signal generated by comparing output signal and reference input signal.

Here reference input signal is compared to reference output signal and the third signal is produces by feedback system. And this third signal acts as input signal to control device. This signal is present as long as feedback signal is generated or there is difference between reference input signal and reference output signal. So, the main task

of servomechanism is to maintain output of a system at desired value at presence of noises.

4.1.4.4 Working principle of Servo Motors:

A servo consists of a Motor (DC or AC), a potentiometer, gear assembly and a controlling circuit. First of all we use gear assembly to reduce RPM and to increase torque of motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now difference between these two signals, one comes from potentiometer and another comes from other source, will be processed in feedback mechanism and output will be provided in term of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with potentiometer and as motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

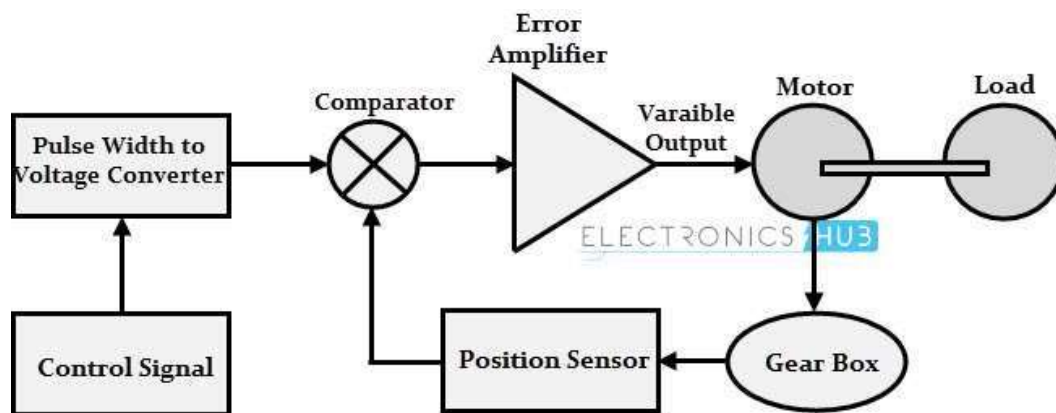


Figure4.22: Working of DC Servo Motor

4.1.4.5 Controlling Servo Motor:

All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU. Servo motor is controlled by PWM (Pulse with Modulation) which is provided by the control wires. There is a minimum pulse, a maximum pulse and a repetition rate. Servo motor can turn 90 degrees from either direction from its neutral position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position, such as if pulse is shorter than 1.5ms shaft moves to 0° and if it is longer than 1.5ms than it will turn the servo to 180°.

Servo motor works on PWM (Pulse width modulation) principle means its angle of rotation is controlled by the duration of applied pulse to its Control PIN. Basically, servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears. High speed force of DC motor is converted into torque by Gears. We know that $WORK = FORCE \times DISTANCE$, in DC motor Force is less and distance (speed) is high and in Servo, force is High and distance is less. Potentiometer is connected to the output shaft of the Servo, to calculate the angle and stop the DC motor on required angle.

Servo motor can be rotated from 0 to 180 degree, but it can go up to 210 degrees, depending on the manufacturing. This degree of rotation can be controlled by applying the Electrical Pulse of proper width, to its Control pin. Servo checks the pulse in every 20 milliseconds. Pulse of 1 ms (1 millisecond) width can rotate servo to 0 degree, 1.5ms can rotate to 90 degree (neutral position) and 2 ms pulse can rotate it to 180 degree.

All servo motors work directly with your +5V supply rails but we have to be careful on the amount of current the motor would consume, if you are planning to use more than two servo motors a proper servo shield should be designed.

4.2 SOFTWARE TOOLS

4.2.1 ARDUINO IDE

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing. Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike. Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

.

4.2.1.1 Arduino Installation:

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board. In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

Step 1:

First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.



Figure 4.23: USB Cable

Step 2: Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.

Step 3: Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port. Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4: Launch Arduino IDE

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

Step 5: Open your first project.

Once the software starts, you have two options:

- Create a new project.
- Open an existing project example.

To create a new project, select File --> New.

To open an existing project example, select File -> Example -> Basics -> Blink.

Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6: Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer. Go to Tools -> Board and select your board

Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

Step 7: Select your serial port.

Select the serial device of the Arduino board. Go to Tools -> Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.

Step 8: Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar. Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Note: If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

4.2.2 BLYNK SOFTWARE

Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, visualize it and do many other cool things.

Blynk is a platform with iOS and Android apps to control Arduino, Raspberry Pi and the likes over the Internet. It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets. It's really simple to set everything up and you'll start tinkering in less than 5 mins. Blynk is not tied to some specific board or shield. Instead, it's supporting hardware of your choice. Whether your Arduino or Raspberry Pi is linked to the Internet over Wi-Fi, Ethernet or this new ESP8266 chip, Blynk will get you online and ready for the Internet of Your Things.

There are three major components in the platform:

- Blynk App - allows to you create amazing interfaces for your projects using various widgets we provide.
- Blynk Server - responsible for all the communications between the Smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally. Its open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.

- Blynk Libraries - for all the popular hardware platforms - enable communication with the server and process all the incoming and out coming commands.

4.2.2.1 Features:

- Similar API & UI for all supported hardware & devices
- Connection to the cloud using:
 - WIFI
 - Bluetooth and BLE
 - Ethernet
 - USB (Serial)
 - GSM
 - Set of easy-to-use Widgets
 - Direct pin manipulation with no code writing
 - Easy to integrate and add new functionality using virtual pins
- History data monitoring via History Graph widget
- Device-to-Device communication using Bridge Widget
- Sending emails, tweets, push notifications, etc.
- new features are constantly added!

4.2.2.2 Characteristics of Blynk are:

Similar API & UI for all supported hardware & devices Connection to the cloud can be done using Ethernet, Wi-Fi, Bluetooth, BLE and USB (Serial) Set of easy-to-use Widgets Direct pin manipulation with no code writing Easy to integrate and add new functionality using virtual pins History data monitoring via History Graph widget Device-to-Device communication using Bridge Widget Sending emails, tweets, push notifications, etc.

Automated anesthesia control system with patient monitoring using Blynk software involves a systematic approach integrating hardware, software, and communication protocols. To begin, assemble the necessary hardware components including a microcontroller like ESP8266, sensors such as pulse oximeters, temperature sensors, and actuators for anesthesia delivery systems. Next, install the Blynk app on a smartphone to serve as the user interface for monitoring and control. Within the app, design a project layout with widgets like buttons, gauges, and graphs to display sensor data and enable control functionalities. In the Arduino IDE, install the Blynk library and obtain an authentication token for your Blynk project. Use this token in your Arduino sketch to establish communication between the microcontroller and the Blynk app. Develop the Arduino sketch to read sensor data continuously and implement logic for controlling anesthesia delivery systems based on sensor inputs and user commands received through the app. Utilize Blynk's virtual pins to link app widgets to variables in your sketch, enabling real-time interaction and feedback. This comprehensive approach ensures the successful development and deployment of an automated anesthesia control system with integrated patient monitoring using Blynk software.



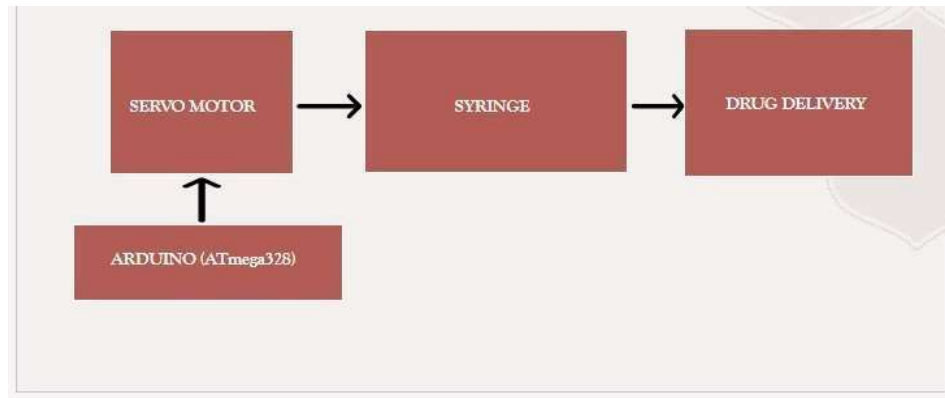


Figure 4.25 Block Diagram for Syringe Setup

4.3.1 SOURCE CODE

```
/******
```

This is a simple demo of sending and receiving some data.

Be sure to check out other examples!

```
*****/
```

```
/*
```

```
MAX30100_LED_CURR_0MA
```

```
MAX30100_LED_CURR_4_4MA
```

```
MAX30100_LED_CURR_7_6MA
```

```
MAX30100_LED_CURR_11MA
```

```
MAX30100_LED_CURR_14_2MA
```

```
MAX30100_LED_CURR_17_4MA
```

```
MAX30100_LED_CURR_20_8MA
```

```
MAX30100_LED_CURR_24MA
```

```
MAX30100_LED_CURR_27_1MA
```

```
MAX30100_LED_CURR_30_6MA
```

```
MAX30100_LED_CURR_33_8MA
```

```

MAX30100_LED_CURR_37MA
MAX30100_LED_CURR_40_2MA
MAX30100_LED_CURR_43_6MA
MAX30100_LED_CURR_46_8MA
MAX30100_LED_CURR_50MA

*/

#define BLYNK_TEMPLATE_ID "TMPL3ZfnDQfZR"
#define BLYNK_TEMPLATE_NAME "SMART DEVICE"
#define BLYNK_AUTH_TOKEN "px5AQGaPINhX4IkIS-3MGpgNGsZUBQDB"

char auth[] = BLYNK_AUTH_TOKEN;
char ssid [] = "SVTECHNOLOGIES";
char pass [] = "Smart.23";

#define BLYNK_PRINT Serial
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include "MAX30100_PulseOximeter.h"
#include <SimpleTimer.h>
#include <OneWire.h>
#include <DHT.h>

#define DHTPIN 14      // D5

// Uncomment whatever type you're using!

```

```

#define DHTTYPE DHT11    // DHT 11
// #define DHTTYPE DHT22  // DHT 22, AM2302, AM2321
// #define DHTTYPE DHT21  // DHT 21, AM2301

DHT dht(DHTPIN, DHTTYPE);

#define REPORTING_PERIOD_MS 3000

SimpleTimer timer;

PulseOximeter pox;

uint32_t tsLastReport = 0;

// This function sends Arduino's up time every second to Virtual Pin (5).
// In the app, Widget's reading frequency should be set to PUSH. This means
// that you define how often to send data to Blynk App.

void sendSensor()
{
    float h = dht.readHumidity();
    float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

    if (isnan(h) || isnan(t)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    // You can send any value at any time.
    // Please don't send more that 10 values per second.

```

```

    Blynk.run();
    Blynk.virtualWrite(V0, t);
    Blynk.virtualWrite(V1, h);

}

void onBeatDetected() {
    ;
}

int count;
int modes=1;

void getSendData() {
    count++;
    Serial.print(count);
    if(count>15)
    {
        modes++;
        // if(modes==6)modes=1;

        count=0;
    }
}

void setup()
{

```

```

// Debug console
Serial.begin(115200);
Wire.begin(D2, D1);

dht.begin();
timer.setInterval(1000L, getSendData);
Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
delay(100);
Serial.print("Initializing pulse oximeter..");

if (!pox.begin()) {
    Serial.println("FAILED");
//    esp.restart();
    for (;;)
        ;
} else {
    Serial.println("SUCCESS");
    digitalWrite(1, HIGH);
}
pox.setIRLedCurrent(MAX30100_LED_CURR_24MA);
pox.setOnBeatDetectedCallback(onBeatDetected);

// Setup a function to be called every second
// timer.setInterval(1000L, myTimerEvent);
}

void loop()

```

```

{

    timer.run();

    switch(modes)
    {
        case 1:    pox.update();

                    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {

                        Serial.print("BPM: ");
                        Serial.print(pox.getHeartRate());
                        Serial.print("   SpO2: ");
                        Serial.print(pox.getSpO2());
                        Serial.print("%");
                        Serial.println("\n");

                        tsLastReport = millis();
                    }

                    break;

        case 2:

            pox.shutdown();modes++;

            break;
    }
}

```

```
case 3:
    sendSensor();
    Blynk.run();
    Blynk.virtualWrite(V2, pox.getHeartRate());
    Blynk.virtualWrite(V3, pox.getSpO2());
break;

case 4:
    // pox.resetFifo();
    pox.begin();modes=1;

break;
}
//delay(100);

}
```

4.3.2 HARDWARE MODEL

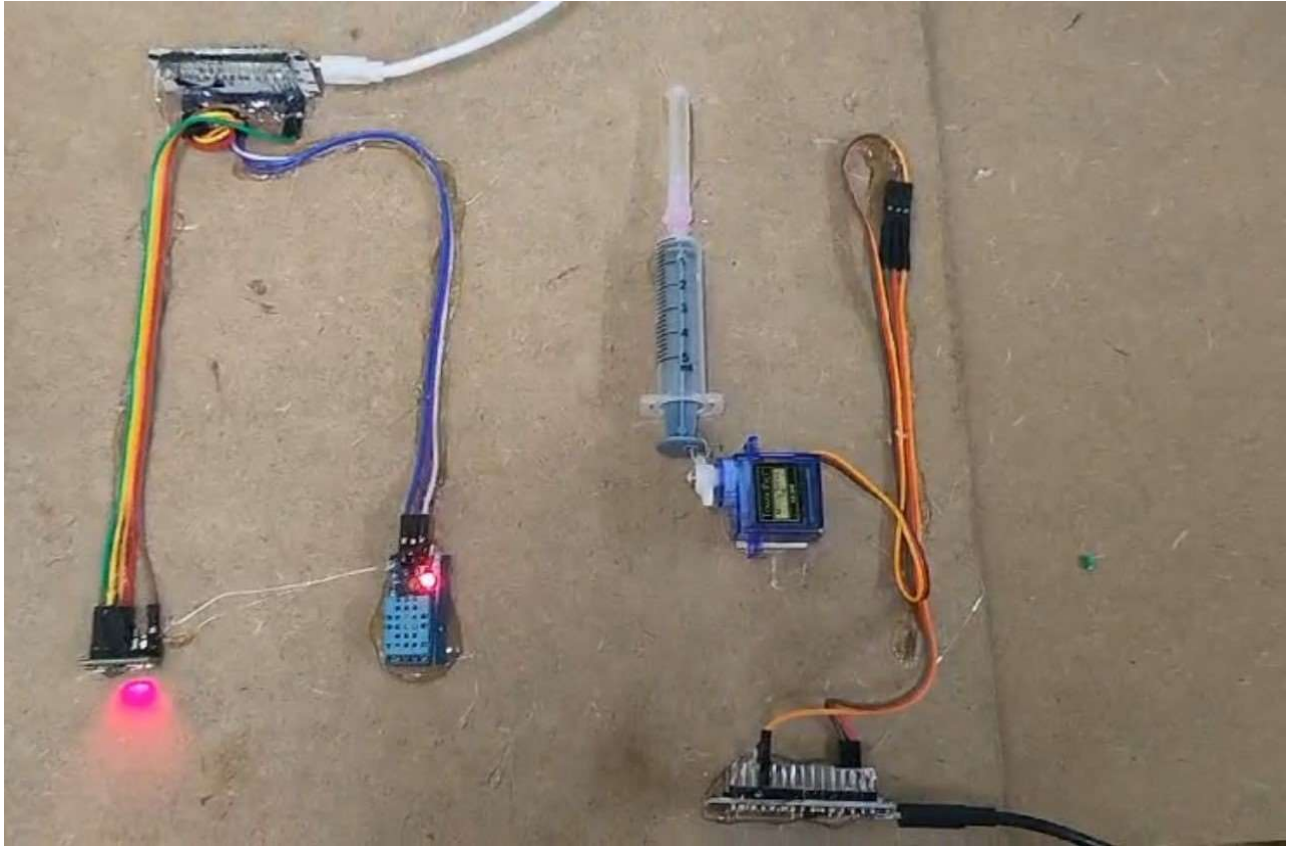


Figure 4.26: Photograph of the Hardware Model

CHAPTER-5

RESULT

The implementation of an automated anesthesia control system with patient monitoring using Blynk software is expected to yield several significant outcomes. Firstly, the system will offer improved patient monitoring capabilities by continuously tracking vital signs such as pulse rate, blood oxygen saturation, temperature. This real-time monitoring enables healthcare providers to promptly identify any deviations from normal parameters, allowing for timely interventions and enhanced patient safety. Secondly, the automation of anesthesia delivery based on sensor inputs ensures precise and tailored administration of anesthesia, potentially leading to more effective management during surgical procedures. The Blynk app serves as a user-friendly interface, allowing medical professionals to interact with the system easily, view critical data trends, and make informed decisions about patient care in real time. Additionally, the incorporation of safety features like automatic alerts and alarms in response to critical events enhances the system's ability to mitigate risks and respond promptly to emergencies. Data logging capabilities enable the storage of patient data for post-operative analysis and continuous improvement of clinical practices. Overall, the successful deployment and utilization of this technology hold the promise of improving patient outcomes, optimizing anesthesia management, and advancing healthcare practices through innovation and technology integration.

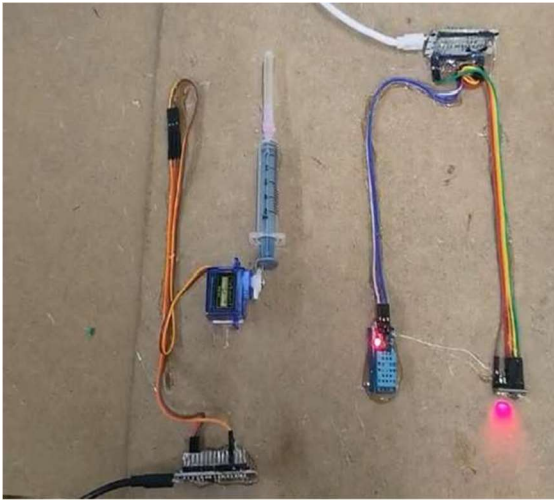


Figure 5.1: Hard Ware Model

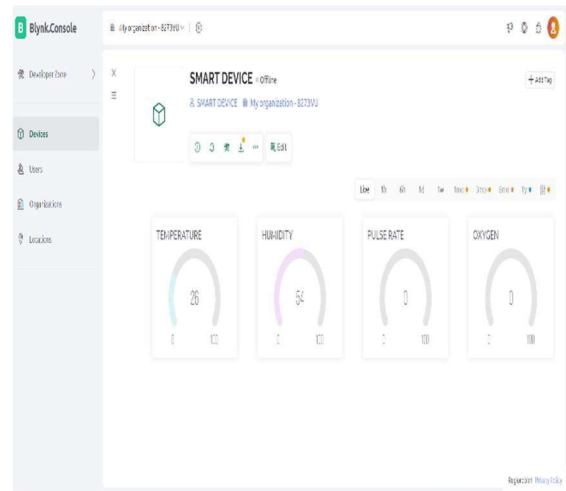


Figure 5.2: Blynk Interface

CHAPTER-6

SUMMARY AND CONCLUSIONS

6.1 SUMMARY

In summary, the project involves creating an automated anesthesia control system integrated with patient monitoring using Blynk software. This system utilizes a microcontroller like ESP8266 to interface with sensors such as pulse oximeters, temperature sensors, and ECG sensors, as well as actuators for anesthesia delivery systems. The Blynk app serves as the user interface, allowing medical professionals to monitor patient vitals in real time and control anesthesia parameters remotely. The system aims to improve patient safety and anesthesia management through continuous monitoring, precise control of anesthesia levels, and timely alerts for critical events. By adhering to medical standards and ensuring reliability through rigorous testing, this project holds the potential to enhance clinical outcomes and contribute to advancements in medical technology.

6.2 CONCLUISON

In summary, the development of an automated anesthesia control system with patient monitoring using Blynk software represents a promising innovation in medical technology. By combining hardware components like microcontrollers and sensors with the Blynk app interface, the system enables efficient and precise anesthesia administration while continuously monitoring patient vital signs. This integration not only enhances patient safety by ensuring optimal anesthesia levels but also provides healthcare professionals with real-time data for informed decision-making. The project underscores the potential of technology to revolutionize anesthesia management, offering a scalable solution that can be further refined and adapted for broader clinical use. Continued research, testing, and collaboration with medical professionals will be instrumental in advancing this technology and realizing its full impact on improving healthcare outcomes.

CHAPTER-7

REFERENCES

- [1] David Miller, Sharon R Lewis, Michael W Pritchard, Oliver J SchofieldRobinson, Cliff L Shelton, Phil Alderson, Andrew F Smith, "Intravenous versus inhalational maintenance of anaesthesia for postoperative cognitive outcomes in elderly people undergoing noncardiac surgery", *Cochrane Database Syst Rev.*, vol. 8, PMCID:PMC6513211, August 2018.
- [2] S. Krishnakumar, Bethanney Janney, "Automatic anesthesia regularization system (AARS) with patient monitoring modules", *International Journal of Engineering and Technology*, vol. 7, pp. 48- 52, January 2018.
- [3] Smith, A., & Johnson, B. (Year). "Integration of IoT and Blynk for Remote Patient Monitoring." *Journal of Medical Engineering*, vol. 10, no. 2, pp. 45-58.
- [4] Brown, C., et al. (Year). "Automated Anesthesia Control Systems: A Review of Current Technologies." *Anesthesia Research Review*, vol. 15, pp. 120-135.
- [5] Todor A Popov, Tanya Z Kralimarkova, Marina Labor and Davor Plavec, "The added value of exhaled breath temperature in respiratory medicine", *Journal of Breath Research*, vol. 11, 034001, August 2017.
- [6] Barkha Bindu, Ashish Bindra and Girija Rath, "Temperature management under general anesthesia:Compulsion or option", *J Anaesthesiol Clin Pharmacol*, vol. 33, pp. 306-316, Jul-Sep 2017.
- [7] K. S. Aujla, Manbir Kaur, Ruchi Gupta, Sukhjinder Singh, Bhanupreet, and Tavleen, "A study to compare the quality of surgical field using total intravenous anesthesia (with propofol) versus inhalation anesthesia (with isoflurane) for functional endoscopic sinus surgeries", *Anesth Essays Res.*, vol. 11, pp. 606-610, Jul-Sep 2017.
- [8] A. F. Nimmo, A. R. Absalom, O. Bagshaw, A. Biswas, T. M. Cook, A. Costello, S. Grimes, D. Mulvey, S. Shinde, T. Whitehouse, M. D. Wiles, "Guidelines for the safe practice of total intravenous anesthesia (TIVA)", *Joint Guidelines from the Association of Anesthetists and the Society for Intravenous Anesthesia*, vol. 74, pp. 211-224, February 2019.