International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

AI-Enabled Smart Monitoring System for Paralysis Patient Using Raspberry Pi-5

[1]G.Yeswanthi, [2]I. Rajshekar, [3]S.Vyshnavi, [4]P.Snehalatha, [5]P.Thanuja, [6]T.M.Vasim Akram

[1,3,4,5,6]UG Student, Dept. of E.C.E., Gates Institute of Technology, Gooty, Anantapur (Dist.) Andhra Pradesh, India
[2]Assistant Professor, Dept., of E.C.E., Gates Institute of Technology, Gooty, Anantapur (Dist.), Andhra Pradesh, India
E-mail: gajulayeswanthi@gmail.com, vyshnavisrisailam@gmail.com, pagireddysnehalath@gmail.com, parimalapeddireddy@gmail.com, shaikwasimakram2003@gmail.com

***Abstract* - Paralysis patients face significant challenges in mobility and communication, making continuous health monitoring essential for their safety and well-being. This project presents an AI-driven real-time paralysis patient monitoring system using Raspberry Pi-5, designed to track health parameters, detect emergencies, and enable assistive communication. The system integrates vital sign sensors, including the DHT11 sensor for temperature and humidity monitoring and a vibration sensor to detect movements or sudden impacts, ensuring early fall detection. A camera module analyse facial expressions and eye-blink detection, allowing non-verbal communication through an AI module. An LED indicator provides real-time status feedback, signaling normal conditions, warnings, or critical alerts. The system employs machine learning models using TensorFlow and OpenCV to analyse distress levels based on facial cues and motion patterns. If abnormal health conditions are detected, an AI-powered alert mechanism triggers a buzzer and instantly sends notifications via the GPRS, SMS, or email, ensuring caregivers receive real-time updates. An SD card stores patient health logs, enabling long-term tracking and analysis. By reducing dependency on caregivers and automating real-time health tracking, this system enhances patient safety and independence. Its affordability, intelligent automation, and AI-powered analytics make it a breakthrough in healthcare technology, offering a scalable and effective solution for paralysis patient monitoring.**

Keywords: AI-driven monitoring, paralysis patients, Raspberry Pi 5, real-time health tracking, assistive communication, DHT11 sensor, vibration sensor, facial expression analysis, eye-blink detection, machine learning, TensorFlow, OpenCV, emergency alerts, GPRS, SMS notifications, buzzer alarm, SD card logging, patient safety, intelligent automation, healthcare technology.

I. INTRODUCTION

Paralysis is a severe medical condition that impairs movement due to nervous system dysfunction. It can arise from various causes, including strokes, spinal cord injuries, brain trauma, or progressive neurological disorders. Depending on its severity, paralysis can limit movement in specific body parts or result in complete immobility. In critical cases, it may also affect essential bodily functions such as breathing, digestion, and speech, requiring continuous medical attention. To overcome these limitations, this project introduces an AI-powered real-time paralysis patient monitoring system using Raspberry Pi 5. The system ensures continuous tracking, distress detection, and automated alerts, improving patient care and emergency response. The key components include: Camera Module – Captures real-time images and videos to analyze the patient's condition. Vibration Sensor – Detects involuntary movements, tremors, or sudden impacts that may indicate distress. DHT11 Sensor – Monitors temperature and humidity, ensuring a comfortable environment. Buzzer & LED Indicators – Provide instant alerts in case of abnormal conditions. SD Card Storage – Logs patient health data for long-term tracking and analysis.

Raspberry Pi Processing Unit – Runs AI-based models to analyze sensor data, identify unusual patterns, and detect emergency situations. GPRS-Enabled GSM Module (SIM800L/SIM900) – Sends automatic SMS notifications and emergency alerts when abnormal conditions are detected. The system employs machine learning algorithms to process real-time data from sensors and the camera module, identifying distress signals such as irregular movements, prolonged inactivity, or unusual facial expressions. Once an abnormal condition is detected, the AI-powered alert mechanism triggers an immediate response: a buzzer alarm activates, LED indicators signal the emergency, and an SMS alert is sent via GPRS to caregivers or healthcare providers. This ensures that critical health updates are communicated instantly, reducing emergency response time and improving patient safety.

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

II. EXISTING SYSTEM

Paralysis patient monitoring has mainly relied on manual supervision by caregivers, where healthcare professionals or family members continuously observe the patient's condition. While this approach provides direct attention, it is not feasible for 24/7 monitoring. Hospitals use bedside monitoring systems to track vital signs like heart rate, temperature, and oxygen levels, but these are expensive, require expert installation, and are mostly limited to hospital settings.

Wearable devices such as smart watches and biosensors offer an alternative by tracking movement and physiological changes. However, they can be uncomfortable, need frequent charging, and may not provide instant emergency alerts. IoT-based monitoring systems integrate sensors with cloud technology, allowing remote tracking, but they depend on internet connectivity, which may be unreliable in some areas.

Machine learning has also been explored for predicting patient movement and detecting emergencies like seizures or falls. However, these methods require high computational power, making them unsuitable for low-power devices like Raspberry Pi. Additionally, most existing solutions focus only on either motion tracking or vital signs, rather than integrating multiple sensors for a complete assessment of the patient's condition. Due to these limitations—high cost, reliance on human supervision, discomfort, and network dependency—there is a need for an AI-driven real-time monitoring system. A multi-sensor approach using Raspberry Pi can provide a cost-effective, automated, and efficient solution for continuous paralysis patient monitoring.

III. PROPOSED SYSTEM

Paralysis significantly affects mobility, making continuous monitoring essential for patient safety and well-being. Patients with paralysis are at risk of involuntary muscle movements, temperature-related discomfort, and distress, which may require immediate attention. To address this challenge, an AI-powered real-time monitoring system is designed using Raspberry Pi and multiple sensors, providing automated assistance and emergency alerts. At the heart of this system is the Raspberry Pi, which acts as the primary processing unit, collecting and analyzing data from various sensors. The system continuously monitors patient conditions and detects abnormalities through AI-powered algorithms.

Key Features and Components:

Vibration Sensor – Detects tremors, involuntary muscle movements, or seizures, which are common in paralysis patients. If abnormal vibrations exceed predefined thresholds, an immediate alert is triggered.

DHT11 Sensor – Continuously tracks temperature and humidity levels around the patient. If environmental conditions become unsuitable, the system sends notifications to caregivers.

Camera Module – Captures real-time video and images to analyze the patient's posture, facial expressions, and body movements. AI algorithms process this data to identify signs of discomfort, distress, or unusual activity.

SD Card Storage – Logs sensor readings and AI analysis results for long-term tracking and medical review. This enables healthcare professionals to study patterns and detect trends in patient health.

Buzzer & LED Indicators – Provide immediate audio-visual alerts in case of emergencies, ensuring that caregivers nearby are notified instantly. Automated Emergency Alerts using GPRS In critical situations, such as severe tremors, abnormal temperature fluctuations, or distress detected via AI, the system automatically sends emergency notifications through multiple communication channels:

GPRS-Enabled GSM Module (SIM800L/SIM900) – Sends real-time SMS alerts directly to caregivers, ensuring they are informed immediately.

Email Notifications – Delivers detailed reports on patient conditions via email for remote caregivers or healthcare professionals.

Automated Call Alerts – The GSM module can trigger emergency calls, allowing caregivers to respond promptly in life-threatening situations.

Block Diagram

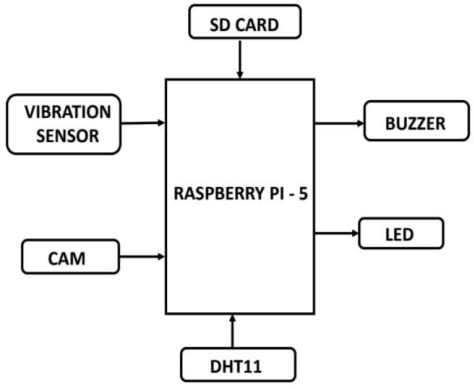


Figure 1: Block Diagram

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

Raspberry Pi – 5

Figure 2: Raspberry Pi – 5

Raspberry Pi is a small single-board Computer developed in UK by Raspberry Pi foundation to promote the teaching of computer science in schools and in developing countries. Original model become far more popular than anticipated sealing outside of its target market, for uses such as robots. The processor at the heart of the Raspberry Pi is a Broadcom BCM28XX. This is the Broadcom System on Chip (SOC) chip use in the Raspberry Pi. The processor from first to third generations include: Raspberry Pi 1: Broadcom BCM2835 SOC with 700MHz CPU speed, L2 cache of 128kb with ARM compatibility AR1176JZF-S (ARMv6) 32-bit RISC ARM. Raspberry Pi 2: Broadcom BCM 2836 SOC with 900MHz CPU speed, L2 cache of 256kb with 32-bit quad-core ARM cortex-A7 (ARMv7). Raspberry Pi 3: Broadcom BCM2837 SOC with 1.2GHz 64-bit quad-core –A53 with 512 kb shared L2 cache (64-bit instruction set ARMv8).

Raspberry Pi Power Supply

Model B+ Power Supply to make the B+ more reliable and actually reduce the current draw, the power supply is completely redesigned.

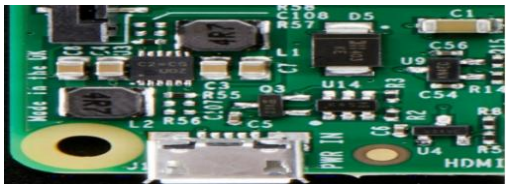


Figure 3: Model B+ Power Supply

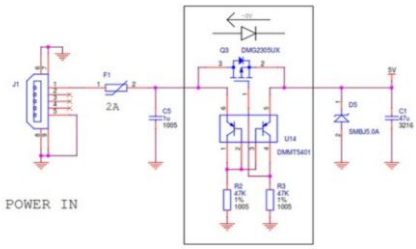


Figure 4: power input protection circuit diagram, likely for overvoltage, overcurrent, and reverse polarity protection using MOSFETs and diodes

There's still the micro USB jack on the left, and the 1A fuse has been upgraded to a 2A fuse. There's also a DMG2305UX (http://adafru.it/dGU) P-Channel MOSFET. These acts as a polarity protection switch but are much lower 'drop-out' than a diode. It has only 52mW resistance so @ 2A its about 0.1V voltage drop. Most diodes would be at least 0.5V. Watch this great video about this technique here: To the right is a protection TVS diode (D5 part #SMBJ5) which protects from over-voltages. So not a lot has changed here (other than putting in a protection FET) There is a PNP-matched-pair action going on around the polarity FET, but its 3AM and I'm not 100% sure what it's for so I'll wait till I get some rest before doing any analysis. Let's look at the 3.3V & 1.8V supplies:

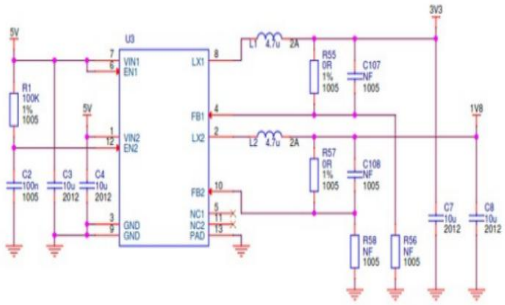


Figure 5: DC-DC buck converter circuit diagram

Instead of heat-spewing LDO (low dropout) regulators, we now have dual buck converters. These are high efficiency converters that can take 5V down to 3.3V or1.8V without as much heat loss. They're more expensive than LDO's but not terribly so! The input to the dual buck is 5V (VIN1 and VIN2) - there's no part number marked here for some reason but it has 12 pins, is a DFN-shaped part (I deal with DFN's all day so I can spot them), and has the marking code C2=CGU0G. with some searching around for a 12-DFN dual buck with 1.8V and 3.3V fixed outputs.

Raspberry Pi Model B+ GPIO Port:

Figure 6: Raspberry Pi model B+ GPIO Port

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

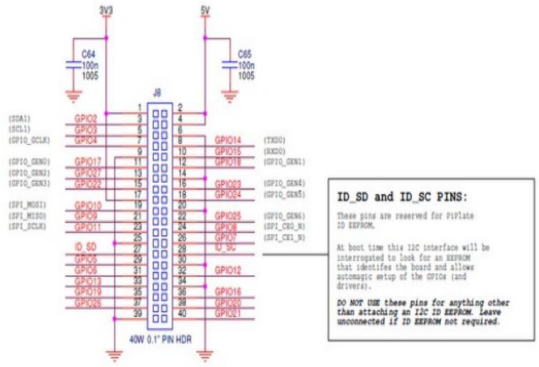


Figure 7: Raspberry Pi 40-pin GPIO header pin out diagram

First thing to notice, the top 26 pins of the 40-pin connector are the same as the original That means that most/many Pi Plates that plug into the Model B will plug into the B+ just fine. They wont sit in the same location - they'll be slid down just a bit but electrically-wise its the same.

Raspberry Pi Configuration

Setting up Raspberry Pi

As said earlier Raspberry Pi comes without any peripheral devices. The first thing to do is to unpack RasPi and protect it with an enclosure (Figure 3). Raspberry Pi can be installed to the protective enclosure without using any tools. The enclosure has plastic clips which are holding the Raspberry Pi in its place.

After Raspberry Pi has been installed to enclosure and well protected, all the necessary peripherals can be attached to it. Just like any other computer, Raspberry Pi needs some basic devices such as display which is connected via the HDMI cable, the mouse and the keyboard, and the internet connection cable. Before plugging the power cable, MicroSD-card should be checked if it is flashed and prepared with an operating system. Also it is recommendable to create a backup folder of the MicroSD-card just in case of complications.

The MicroSD-card can be checked with a card-reader. The card-reader can be found from most of the laptops and desktop computers. Insert the MicroSD-card into the card-reader and check that there is something stored in the MicroSD-card. If everything looks good, take the MicroSD-card and plug it into the Raspberry Pi. Now the power cable can be connected.

Raspberry Pi does not have any kind of power switch so it will start up immediately when the power cable is connected to it. At the start up text starts to flow on the monitor and shortly after that there appears a configuration menu. The configuration menu is called Raspi-config (Figure 4). In Raspi-config it is possible to change some of the settings on Raspberry.

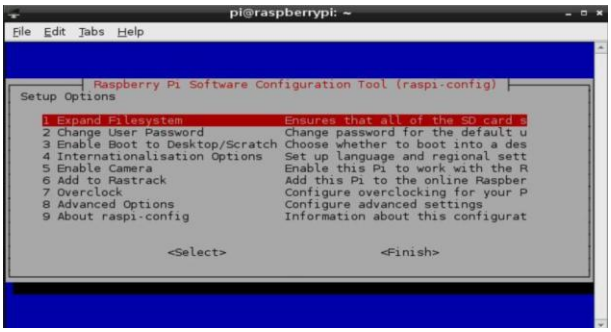


Figure 8: The most important settings that should be checked in Raspi-config

Expand File system, where it is necessary to check that Raspi can use the whole memory capacity of the MicroSD card. Otherwise the memory can run out fast. Internationalisation Options, where it is possible to choose between different languages and the time zones. Advanced Options, if the internet cable has been plugged in, it is possible to update Raspi to the latest version available. (McManus, S. & Cook, M. 2013, 38.) It is recommendable that users who do not have so much experience with Linux operating systems should choose the English language because then help and advice can be found more easily from the internet. It is possible to get back to the Raspi-config and change the settings also after the first setup by typing the following command into the terminal:

sudo raspi-config

After making the changes on the Raspberry Pi's settings, the settings can be accepted by choosing the Finishoption. Now the terminal view should appear and it might be asking for the username and the password. The username in Raspian Wheezy is by default pi and the password should be raspberry. Notice that these are written in small letters. The Linux is letter case sensitive and it will recognize the difference between small and capital letters. The next step is logging in to Raspberry and instead of the graphical environment there will be a command console flashing. However, the graphical environment, or so called desktop view, can be started by entering the command:

Startx

Now Raspberry will be loading for a while and a few seconds later there will ap-pear a more user friendly desktop view. It is recommended to learn how to use the command console as it makes some of the actions faster than doing them

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

in the desktop view. So far the basic configurations are made for the Raspberry. There might still be some things that are not working correctly. For instance, the keyboard layout might be defined to be in UK style which is the default keyboard layout setting on Rasp-berry Pi. This can be frustrating and annoying. The layout can be changed easily by opening the LXTerminal which opens the command console. Open the key-board file in the command console with the nano text editor by typing the following command: sudo nano /etc/default/keyboard the keyboard configuration file (Figure 5) will appear and it can be modified. The keyboard layout can be changed by replacing the XKBLAYOUT value as shown in Figure 5. After the file is edited it can be saved by pressing CTRL + O key combi-nation.

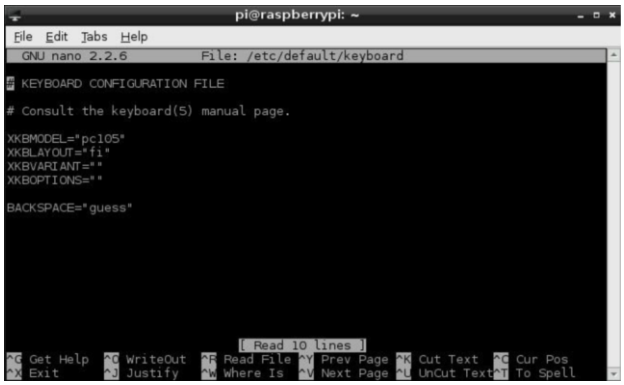


Figure 9: The keyboard configuration, setting the layout to the Finnish "fi"

Controlling the GPIO pins with Python This chapter discovers the GPIO connector and how it canbe used in controlling. The first experiments with the GPIO were to light up a LED (light-emitting diode) through the Python Shell. The second experiment is little bit more complex and it demonstrates the control-ling loop of an heating element. The heating element will start to heat the room when room's temperature is getting below the pre-defined lower limit and stops heating when the temperature in the room reaches the second pre-defined upper limit.

Controlling the LED with the GPIO

This is the first experiment with the GPIO connector and it demonstrates how to use it in controlling. This experiment requires a LED and a resistor. The resistor's resistance can be calculated from the Ohm's law which is shown in Formula 1. Defining the resistance from the Ohm's law:

$$U = R * I \quad (1)$$

$$R = \frac{U}{I} = \frac{3.3V_{GPIO\ voltage} - 1.18V_{LED\ voltage\ drop}}{10mA_{LED\ current}} = 212\,\Omega \quad (2)$$

Where,

U is voltage
R is resistance
I is current

The resistors above 212 Ω are suitable and can be used for lightning the LED directly from the GPIO.

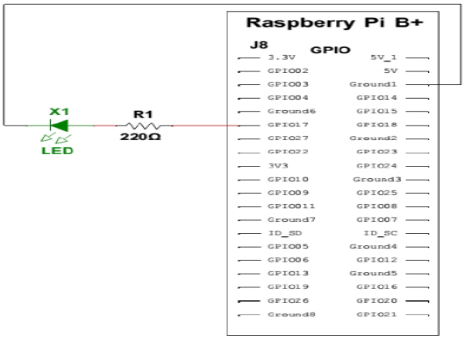


Figure 11: The wiring for the LED and resistor

After the wirings are done the Python library called python-rpi.gpio needs to be installed. This library allows controlling the GPIO pins. It can be installed with the following command: sudo apt-get install python-rpi.gpio When the installation is finished, open up a Python Shell from the terminal as root user and import the RPi.GPIO library.

import RPi.GPIO as GPIO

Next thing to do is to set the mode to use the pin numbers from the ribbon cable board and define one of the GPIO pins to be an output. For instance the GPIO 17:

GPIO.setmode(GPIO.BCM) # Ribbon cable board

GPIO.setup(17, GPIO.OUT) # Defines the GPIO17 to be output

Now it is possible to control the GPIO17 pin to high and low. The LED will light up when the pin 17 is set to high and when it is set to low the LED will turn off.

GPIO.output(17, GPIO.HIGH) # Turns the GPIO17 to high

GPIO.output(17, GPIO.LOW) # Turns the GPIO17 to low

Installing the Pi NoIR camera module

The Raspberry Pi's NoIR camera module board comes in anti-static plastic bag. It is fast and easy to install. The camera module can be mounted to the protective case's cover, where is reserved slot for the camera. (Figure 12) It is screwed with two small screws, and the ribbon cable is connected to the

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

Raspberry Pi's camera connection port. The connection port is located between the 3.5mm audio jack and the HDMI socket. The connection port's clip has to be pulled up before plugging the camera module's ribbon cable on its place after mounting the camera module, it is required to enable the camera module from the Raspi-config configuration tool and then Raspberry Pi has to be rebooted so that the changes will take effect.

Creating a Python script for taking pictures

First things to consider before creating the script which takes the picture and stores it automatically are: where the picture is stored, finding the right parameters for the picture so that the image quality and size does not suffer too much.

After a while, some limitations for the pictures are found. The size and quality are reduced to minimize the picture size on the hard drive. The Quality of 75% and the resolution of 1280x720 pixels are sufficient. With these parameters the picture size on the hard drive is around 500KB. That is good starting point, and trade-off be-tween picture quality and available space for picture saving.

All the pictures which are taken by the Python script will be saved to the own folder with current timestamp filename. The folder is located at /var/www/camera/. Apache2 is hosting the folder so that the pictures are available on the website.

Creating the script starts with placing the shebang information and importing the necessary libraries. These libraries are datetime, picamera and time.

```
#!/usr/bin/env/ python

import datetime

import picamera

import time
```

On the second step a function called take Picture should be defined. It does not take any input variables. The function consists of three parts. The first part is the general settings, where the location to the saved pictures and the filename are defined.

```
def takePicture():

location="/var/www/camera/" #Location to the files

date=datetime.datetime.now() #Get current date

file_name=date.strftime("%Y-%m-%d %H%M") #Format the string
```

The second part of the function is defining the settings for the picture size and it starts also the preview mode.

```
#configuration for the pictures

camera = picamera.PiCamera()

camera.resolution = (1280,720)

camera.start_preview()
```

In the last part of the function, the preview mode is kept on for a certain time to warm up the camera. After the warm up time, the function captures the picture and saves it to the predefined location. The picture is named with current timestamp. At the end of the script the preview mode is stopped and the camera is closed.

```
time.sleep(2) #Camera warm up time

# Capture the picture and saved it with the current date

camera.capture("%s%s.jpg" % (location,file_name), quality=75)

camera.stop_preview()

camera.close()
```

Architecture

ARM vs. x86

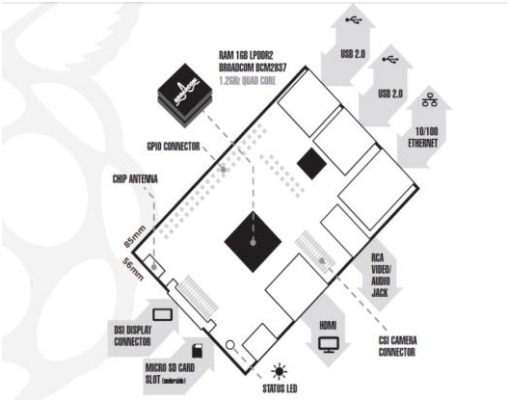


Figure 12: Raspberry Pi board layout diagram

The Raspberry Pi is powered by the Broadcom BCM2837 SoC, which integrates the CPU, GPU, audio, and communication hardware into a single unit located beneath the 256MB Hynix memory chip. This system-on-chip (SoC) design makes it compact and efficient.

Unlike traditional desktop processors, the BCM2837 operates on the ARMv6 architecture, which is optimized for

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

low power consumption. This design enables the Raspberry Pi to function with just a 5V 1A power supply via micro-USB, producing minimal heat and eliminating the need for cooling fans or heat sinks.

However, because it is ARM-based, the Raspberry Pi cannot run x86 desktop software designed for Intel or AMD processors. Additionally, it does not support programs built for the more advanced ARMv7 architecture, though software can often be adapted for ARMv6. Despite this, there is extensive software compatibility for Raspberry Pi, and its growing popularity continues to expand the range of available applications.

Getting Started with the Raspberry Pi

Now that you have a understanding of how the Pi differs from other computing devices, it's time to get started. If you've just received your Pi, take it out of its protective anti-static bag and place it on a flat, non-conductive surface before continuing with this chapter.

Connecting a Display

Before you can start using your Raspberry Pi, you're going to need to connect a display. The Pi supports three different video outputs: composite video, HDMI video and DSI video. Composite video and HDMI video are readily accessible to the end user, as described in this section, while DSI video requires some specialised hardware.

Composite Video

Composite video, available via the yellow-and-silver port at the top of the Pi known as an RCA phono connector is designed for connecting the Raspberry Pi to older display devices. As the name suggests, the connector creates a composite of the colours found within an image—red, green and blue—and sends it down a single wire to the display device, typically an old cathode-ray tube (CRT) TV. The yellow RCA phono connector, for composite video output When no other display device is available, a composite video connection will get you started with the Pi. The quality, however, isn't great. Composite video connections are significantly more prone to interference, lack clarity and run at a limited resolution, meaning that you can fit fewer icons and lines of text on the screen at once.

HDMI Video Provides high-definition (1920x1080) display for monitors and HDTVs.

Ensures pixel-perfect picture quality with a high-speed digital connection.

Compatible with DVI-D monitors using an HDMI-to-DVI cable.

Not compatible with VGA monitors without an expensive adapter.

DSI Video

Uses a small ribbon connector above the SD card slot.

Supports Display Serial Interface (DSI) displays, commonly used in tablets and smartphones.

Rarely available for retail; mainly for engineers and embedded systems.

Beginners should use HDMI for an easier setup.

Connecting Audio on Raspberry Pi

HDMI Audio – If using HDMI, audio is transmitted with video. Just connect the cable. DVI-D & Composite Video – No audio via DVI; use the 3.5mm audio jack for sound. Headphones & Speakers – Direct headphone connection works but may be low volume; powered speakers recommended. Adapters & Cables – Use a 3.5mm to RCA cable for TVs/amplifiers or 3.5mm to 3.5mm for PC speakers.

If you want to reduce the number of power sockets in use, connect the Raspberry Pi's USB power lead to your powered USB hub. This way, the Pi can draw its power directly from the hub, rather than needing its own dedicated power socket and mains adapter. This will only work on hubs with a power supply capable of providing 700mA to the Pi's USB port, along with whatever power is required by other peripherals.

Connecting the keyboard and mouse is as simple as plugging them in to the USB ports, either directly in the case of a Model B or via a USB hub in the case of a Model A Note on Storage

As you've probably noticed, the Raspberry Pi doesn't have a traditional hard drive. Instead it uses a Secure Digital (SD) memory card, a solid-state storage system typically used in digital cameras. Almost any SD card will work with the Raspberry Pi, but because it holds the entire operating system, it isnecessary for the card to be at least 2 GB in capacity to store all the required files.

SD cards with the operating system preloaded are available from the official Raspberry Pi Store along with numerous other sites on the Internet. If you've purchased one of these, or received it in a bundle with your Pi, you can simply plug it in to the SD card slot on the bottom side of the

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

left-hand edge. If not, you'll need to install an operating system—known as flashing—onto the card before it's ready to go.

Some SD cards work better than others, with some models refusing to work at all with the Raspberry Pi. For an up-to-date list of SD card models known to work with the Pi, visit the eLinux

Flashing the SD Card

To prepare a blank SD card for use with the Raspberry Pi, you'll need to flash an operating system onto the card.

During the following, you'll be using a software utility called dd. Used incorrectly dd will happily write the image to your main hard drive, erasingyour operating system and all your stored data. Make sure you read the instructions in each section thoroughly and note the device address of yourSD card carefully. Read twice, write once!

Flashing from Linux

If your current PC is running a variant of Linux already, you can use the dd command to write the contents of the image file out to the SD card. This is a text-interface program operated from the command prompt, known as a terminal in Linux parlance.

Follow these steps to flash the SD card:

1. Open a terminal from your distribution's applications menu.

2. Plug your blank SD card into a card reader connected to the PC.

3. Type sudo fdisk -l to see a list of disks. Find the SD card by its size, and note the device address (/dev/sdX, where X is a letter identifying the storage device. Some systems with integrated SD card readers may use the alternative format /dev/mmcblkX—if this is the case, remember to change the target in the following instructions accordingly).

4. Use cd to change to the directory with the .img file you extracted from the Zip archive.

5. Type sudo dd if=imagefilename.img of=/dev/sdX bs=2M to write the file imagefilename.img to the SD card connected to the device address from step 3. Replace imagefilename.img with the actual name of the file extracted from the Zip archive. This step takes a while, sobe patient! During flashing, nothing will be shown on the screen until the process is fully complete. Figure 1-5: Flashing the SD card using the dd command in Linux.

Flashing from OS X

If your current PC is a Mac running Apple OS X, you'll be pleased to hear that things are as simple as with Linux. Thanks to a similar ancestry, OS X and Linux both contain the dd utility, which you can use to flash the system image to your blank SD card as follows:

1. Select Utilities from the Application menu, and then click on the Terminal application.

2. Plug your blank SD card into a card reader connected to the Mac.

3. Type diskutil list to see a list of disks. Find the SD card by its size, and note the device address (/dev/diskX, where X is a letter identifying the storage device).

4. If the SD card has been automatically mounted and is displayed on the desktop, type diskutil unmountdisk /dev/diskX to unmount it before proceeding.

5. Use cd to change to the directory with the .img file you extracted from the Zip archive.

6. Type dd if=imagefilename.img of=/dev/diskX bs=2M to write the file imagefilename.img to the SD card connected to the device address from step 3. Replace imagefilename.img with the actual name of the file extracted from the Zip archive.This step takes a while, so be patient!

Connecting External Storage

The Raspberry Pi uses an SD card as its primary storage, but space may be limited. Instead, you can connect USB storage devices like: Flash DrivesExternal Hard DrivesSSDsTo use them, they must be mounted

Connecting the Network

Model B – Has built-in networking.

Model A – Requires a USB network adapter for connectivity.

Wired Networking

To connect your Raspberry Pi to a network, use an RJ45 Ethernet cable and plug it into a router, switch, or hub. If no router is available, you can directly connect the Pi to a PC/laptop with a standard Ethernet cable.

Key Features & Setup:

Auto-MDI (Auto Crossover) – No need for a special crossover cable; any RJ45 cable works.

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

DHCP Support – The Pi automatically gets an IP address when connected to a DHCP-enabled network.

Manual Configuration – If no DHCP is available, the Pi requires manual IP setup (covered in Chapter 4).

Internet Access – Direct PC-to-Pi connections don't provide Internet unless network bridging is enabled.

Wireless Networking

Current Raspberry Pi models don't have built-in Wi-Fi, but you can use a USB wireless adapter to connect to networks, including 802.11n high-speed Wi-Fi.

Connecting Power

Uses a micro-USB port (same as most smartphones). Requires 700mA; some chargers may not provide enough power. Avoid powering from a PC's USB port as it may cause issues. No power button – Pi starts when plugged in and stops when unplugged.

The GPIO Port

General-Purpose Input/Output (GPIO) has 26 pins (arranged in two rows). Standard 2.54mm pin spacing (compatible with breadboards). Each pin has a specific function—important for circuits & hardware projects. Be careful with pin numbering, as it differs from other devices.

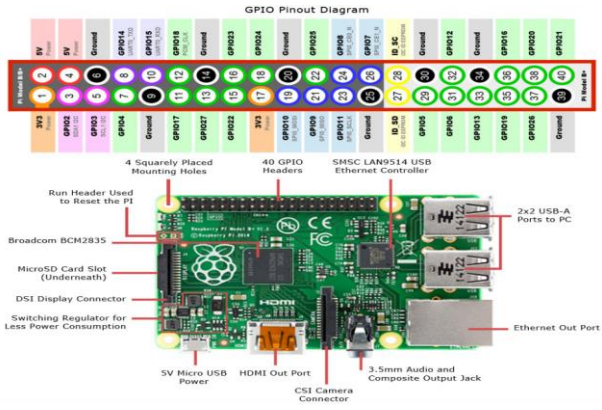


Figure 13: Raspberry Pi GPIO pinout diagram

Although the Pi's GPIO port provides a 5 V power supply, tapped from the incoming power on the micro-USB hub, on Pin 2, the Pi's internal workings are based on 3.3 V logic. This means that the components on the Pi work from a 3.3 V power supply. If you're planning on creating a circuit that will interface with the Pi through its GPIO port, make sure you are using components compatible with 3.3 V logic or are passing the circuit through a voltage regulator before it reaches the Pi.

Connecting a 5 V supply to any pin on the Raspberry Pi's GPIO port, or directly shorting either of the power supply pins (Pin 1 and Pin 2) to any other pin will result in damage to the Pi. Because the port is wired directly to pins on the Broadcom BCM2837 SoC processor, you will not be able to repair any damage you do to it. Always be extra careful when working around the GPIO port.

The GPIO port provides seven pins for general-purpose use by default: Pin 11, Pin 12, Pin 13, Pin 15, Pin 16, Pin 18 and Pin 22. Additionally, Pin 7—while defaulting to providing a clock signal for general purpose use—can also be used as a general purpose pin, giving eight pins in total. These pins can be toggled between two states: high, where they are providing a positive voltage of 3.3 V; and low, where they are equal to ground or 0 V. This equates to the 1 and 0 of binary logic, and can be used to turn other components on or off. You'll learn more about this later in the chapter.

The Pi's internal logic operates at 3.3 V. This is in contrast to many common microcontroller devices, such as the popular Arduino and its variants, which typically operate at 5 V. Devices designed for the Arduino may not work with the Pi unless a level translator or optical isolator is used between the two. Likewise, connecting pins on a 5 V microcontroller directly to the Raspberry Pi's GPIO port will not work and may permanently damage the Pi.

In addition to these general-purpose pins, the GPIO port has pins dedicated to particular buses. These buses are described in the following subsections.

Installing the GPIO Python Library

Since the launch of the Pi, numerous developers have created software modules known as libraries for making full use of its various functions. In particular, programmers have addressed the Pi users' need to access the GPIO port without having to know low-level programming.

These libraries are designed to extend the functionality of the base Python language, much like the pygame software described, "Python Basics". Installing one of these libraries gives Python the ability to easily address the Pi's GPIO port, although it means that anyone planning to use the software you create will also have to download and install the library before it will work. There are several GPIO Python libraries available, but for the purpose of this section, we recommend that you use the raspberry-gpio-python library, which was at version 0.2.0 at the time of writing.

Although it's possible to download the Python library through a web browser, it's significantly quicker to do so

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

through the terminal as part of the installation process. Just follow these steps:

1. Open a terminal window on your Raspberry Pi from the Accessories menu, or use the console if you haven't loaded a desktop environment.

2. Type wget http://raspberry-gpio-python.googlecode.com/files/RPi.GPIO-0.2.0.tar.gz to download the library to your home directory. If a newer version has been released, replace the version number—0.2.0—with the current version.

3. Type tar xvzf RPi.GPIO-0.2.0.tar.gz to extract the contents of the file. This command is case-sensitive, so make sure to type the capital letters.

4. Type cd RPi.GPIO-0.2.0 to change to the newly created directory. Again, if you downloaded a newer version of the library, replace the version number with that of the downloaded version.

5. Type sudo python setup.py install to install the library into Python. Although the GPIO library is now installed in Python, it won't be loaded by default. Like pygame, the library needs to be explicitly imported into your program m. To use the library, start your program with import RPi.GPIO as GPIO at the top. You'll learn more about this in the following examples.

The Raspberry Pi's GPIO port does not provide any protection against voltage spikes or electrical shorts. Always make sure you've checked that your circuit is sound before connecting it to the Pi. If possible, use an isolation board such as the Gertboard to provide protection.

Calculating Limiting Resistor Values An LED needs a current limiting resistor to protect it from burning out. Without a resistor, an LED will likely only work for a short time before failing and available value is 68 Ω, which will adequately protect the LED. If you don't know the forward voltage and forward current of your LEDs (for example, if the LEDs did not come with documentation or were salvaged from scrap electronics), err on the side of caution and fit a reasonably large resistor. If the LED is too dim, you can revise downwards—but it's impossible to repair an LED that has been blown.

GPIO Output: Flashing an LED

For the first example, you'll need to build a simple circuit consisting of an LED and a resistor. The LED will provide visual confirmation that the Pi's GPIO port is doing what your Python program tells it to do, and the resistor will limit the current drawn by the LED to protect it from burning out. To assemble the circuit, you'll need a breadboard, two jumper wires, an LED and an appropriate current-limiting resistor (as described in the "Calculating Limiting Resistor Values" sidebar). Although it's possible to assemble the circuit without a breadboard by twisting wires together, a breadboard is a sound investment and makes assembling and disassembling prototype circuits straightforward. Assuming the use of a breadboard, assemble the circuit in the following manner to match

1. Insert the LED into the breadboard so that the long leg (the anode) is in one row and the shorter leg (the cathode) is in another. If you put the LED's legs into the same row, it won't work.

2. Insert one leg of the resistor into the same row as the LED's shorter leg, and the other resistor leg into an empty row. The direction in which the resistor's legs are placed doesn't matter, as a resistor is a non-polarised (direction-insensitive) device.

3. Using a jumper wire, connect Pin 11 of the Raspberry Pi's GPIO port (or the corresponding pin on an interface board connected to the GPIO port) to the same row as the long leg of the LED.

4. Using another jumper wire, connect Pin 6 of the Raspberry Pi's GPIO port (or the corresponding pin on an interface board connected to the GPIO port) to the row that contains only one leg of the resistor and none of the LED's legs.

Be very careful when connecting wires to the Raspberry Pi's GPIO port. As discussed earlier in the chapter, you may do serious damage to the Pi ifyou connect the wrong pins.

A breadboard circuit for a simple LED output At this point, nothing will happen. That's perfectly normal: by default, the Raspberry Pi's GPIO pins are switched off. If you want to check your circuit immediately, move the wire from Pin 11 to Pin 1 to make the LED light up. Be careful not to connect it to Pin 2, though: a current-limiting resistor suitable for a 3.3 V power supply will be inadequate to protect the LED when connected to 5 V. Remember to move the wire back to Pin 11 before continuing. To make the LED do something useful, start a new Python project. "An Introduction to Python", you can use a plain text editor or the IDLE software included in the recommended Debian distribution for this project as well. Before you can use the GPIO library you installed earlier in this chapter, you'll need to import it into your Python project. Accordingly, start the file with the following line:

import RPi.GPIO as GPIO

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

Remember that Python is case-sensitive, so be sure to type RPi.GPIO exactly as it appears. To allow Python to understand the concept of time (in other words, to make the LED blink, rather than just turning it on and off), you'll also need to import the time module. Add the following line to the project:

import time

With the libraries imported, it's time to address the GPIO ports. The GPIO library makes it easy to address the general-purpose ports through the instructions GPIO.output and GPIO.input, but before you can use them, you'll need to initialise the pins as either inputs or outputs. In this example, Pin 11 is an output, so add the following line to the project:

GPIO.setup(11, GPIO.OUT)

This tells the GPIO library that Pin 11 on the Raspberry Pi's GPIO port should be set up as an output. If you were controlling additional devices, you could add more

GPIO.setup

lines into the project. For now, however, one will suffice.

With the pin configured as an output, you can switch its 3.3 V supply on and off in a simple demonstration of binary logic. The instruction

GPIO.output(11, True)

will turn the pin on, while

GPIO.output(11, False)

switches it off again. The pin will remember its last state, so if you only give the command to turn the pin on and then exit your Python program, the pin will remain on until told otherwise.

Although you could just add GPIO.output(11, True) to the Python project to switch the pin on, it's more interesting to make it blink. First, add the following line to create an infinite loop in the program:

while True:

Next, add the following lines to switch the pin on, wait 2 seconds, and then switch it off again before waiting another 2 seconds. Make sure each line starts with four spaces, to signify that it is part of the infinite while loop:

GPIO.output(11, True)
time.sleep(2)
GPIO.output(11, False)
time.sleep(2)

The finished program should look like this (see Figure 12-4):

import RPi.GPIO as GPIO
import time
GPIO.setup(11, GPIO.OUT)
while True:
GPIO.output(11, True)
time.sleep(2)
GPIO.output(11, False)
time.sleep(2)

The gpiooutput.py program, being edited in nano, and waiting for its final line Save the file as gpiooutput.py. If you're using a Python development environment such as SPE, don't try to run the program from within the editor. Most Raspberry Pi Linux distributions restrict the use of the GPIO port to the root user, so the program will need to be run using the command sudo python gpiooutput.py at the terminal to get it started. If all has gone well, you should see the LED begin to blink on and off at regular intervals—and you've created your first home-made output device for the Pi.

If things don't work, don't panic. First, check all your connections. The holes in a breadboard are quite small, and it's easy to think you've inserted a component into one row only to find it's actually in another. Next, check that you've connected the circuit to the right pins on the GPIO port—with no labelling on the Pi itself, mistakes are unfortunately easy to make. Finally, doublecheck your components—if the forward voltage of your LED is higher than 3.3 V or if your current limiting resistor is too large, the LED won't light up. Although this example is basic, it's a good demonstration of some fundamental concepts. To extend its functionality, the LED could be replaced with a buzzer to make an audible alert, or a servo or motor as part of a robotics platform. The code used to activate and deactivate the GPIO pin can be integrated into other programs, causing an LED to come on when new email arrives or a flag to be raised when a friend has joined an IRC channel.

CAM

A webcam is a video camera that feeds or streams an image or video in real time to or through a computer to a computer network, such as the Internet. Webcams are typically small cameras that sit on a desk, attach to a user's monitor, or are built into the hardware. Webcams can be used during a video chat session involving two or more people, with conversations that include live audio and video. For

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

example, Apple's iSight camera, which is built into Apple laptops, iMacs and a number of iPhones, can be used for video chat sessions, using the iChat instant messaging program (now called Messages). Webcam software enables users to record a video or stream the video on the Internet. As video streaming over the Internet requires a lot of bandwidth, such streams usually use compressed formats. The maximum resolution of a webcam is also lower than most handheld video cameras, as higher resolutions would be reduced during transmission. The lower resolution enables webcams to be relatively inexpensive compared to most video cameras, but the effect is adequate for video chat sessions.

Figure 14: CAM

Various lenses are available, the most common in consumer-grade webcams being a plastic lens that can be manually moved in and out to focus the camera. Fixed-focus lenses, which have no provision for adjustment, are also available. As a camera system's depth of field is greater for small image formats and is greater for lenses with a large f-number (small aperture), the systems used in webcams have a sufficiently large depth of field that the use of a fixed-focus lens does not impact image sharpness to a great extent. Most models use simple, focal-free optics (fixed focus, factory-set for the usual distance from the monitor to which it is fastened to the user) or manual focus.

Interface

Typical interfaces used by articles marketed as a "webcam" are USB, Ethernet and IEEE (denominated as IP camera). Further interfaces such as e.g. Composite video or S-Video are also available

The USB video device class (UVC) specification allows inter-connectivity of webcams to computers without the need for proprietary device drivers.

Software

Various proprietary as well as free and open-source software is available to handle the UVC stream. One could use Guvcview or GStreamer and GStreamer-based software to handle the UVC stream.

Dht11 Sensor:

This sensor is used here to monitor the humidity variation of the environment where the crops are cultivated. This is a digital sensor and measures the humidity value in percentage format.DHT11 humidity and temperature sensor is available as a sensor and as a module. The difference between this sensor and module is the pull-up resistor and a power-on LED. DHT11 is a relative humidity sensor. To measure the surrounding air this sensor uses a thermistor and a capacitive humidity sensor.

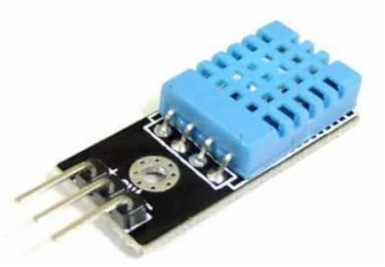


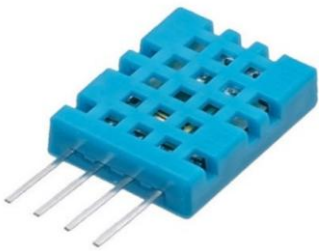
Figure 15: DHT11 Sensor

DHT11 is a low-cost digital sensor for sensing temperature and humidity. This sensor can be easily interfaced with any micro-controller such as Arduino, Raspberry Pi etc… to measure humidity and temperature instantaneously.

Working

The DHT11 sensor measures temperature and humidity using a capacitive humidity sensor and a Negative Temperature Coefficient (NTC) thermistor. The humidity sensor changes capacitance based on moisture levels, while the thermistor's resistance decreases as temperature increases.

Temperature Range: 0 to 50°C (±2°C accuracy) Humidity Range: 20% to 80% (±5% accuracy)Sampling Rate: 1Hz (one reading per second) Operating Voltage: 3V to 5V Current Consumption: Max 2.5mA Pins: VCC, GND, Data, NC (Not Connected) Pull-up Resistor: 5kΩ–10kΩ for communication.

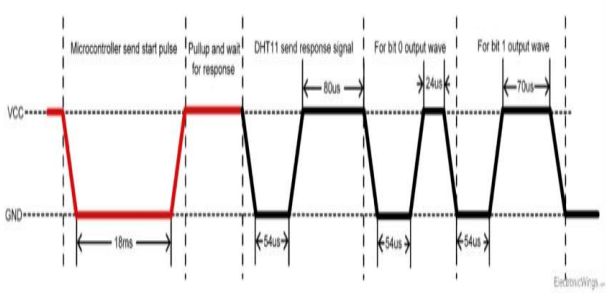


International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

Humidity (DHT11):

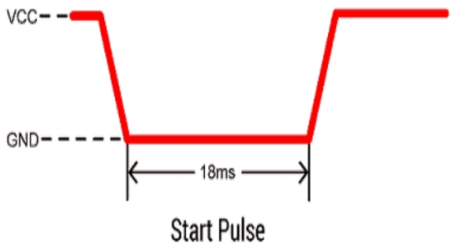
- DHT11 uses only one wire for communication. The voltage levels with certain time value defines the logic one or logic zero on this pin.

- The communication process is divided in three steps, first is to send request to DHT11 sensor then sensor will send response pulse and then it starts sending data of total 40 bits to the microcontroller.



Communication process

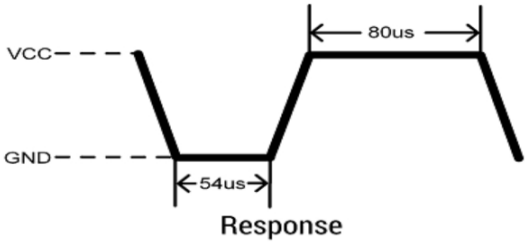
Start pulse (Request)



To start communication with DHT11, first we should send the start pulse to the DHT11 sensor.

To provide start pulse, pull down (low) the data pin minimum 18ms and then pull up, as shown in diag.

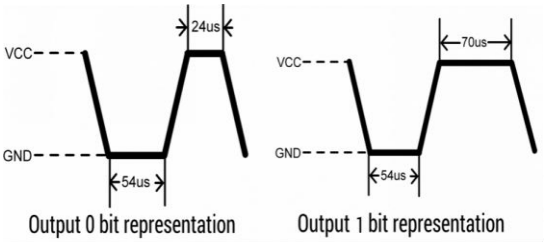
Response



After getting start pulse from, DHT11 sensor sends the response pulse which indicates that DHT11 received start pulse.

The response pulse is low for 54us and then goes high for 80us.

Data



After sending the response pulse, DHT11 sensor sends the data, which contains humidity and temperature value along with checksum. The data frame is of total 40 bits long, it contains 5 segments (byte) and each segment is 8-bit long.

In these 5 segments, first two segments contain humidity value in decimal integer form. This value gives us Relative Percentage Humidity. 1st 8-bits are integer part and next 8 bits are fractional part.

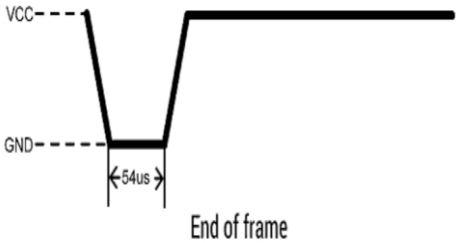
Next two segments contain temperature value in decimal integer form. This value gives us temperature in Celsius form.

Last segment is the checksum which holds checksum of first four segments.

Here checksum byte is direct addition of humidity and temperature value. And we can verify it, whether it is same as checksum value or not. If it is not equal, then there is some error in the received data.

Once data received, DHT11 pin goes in low power consumption mode till next start pulse.

End of frame



After sending 40-bit data, DHT11 sensor sends 54us low level and then goes high. After this DHT11 goes in sleep mode.

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

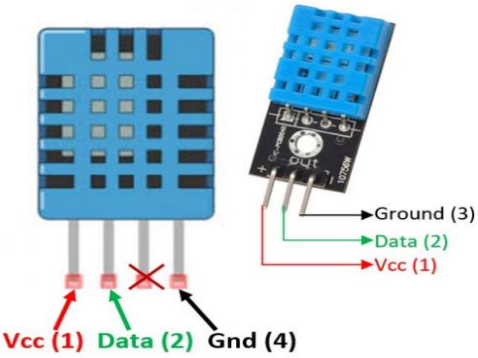


Figure 16: DHT11 temperature and humidity sensor pinout diagram

Vibration Sensor (SW-420)

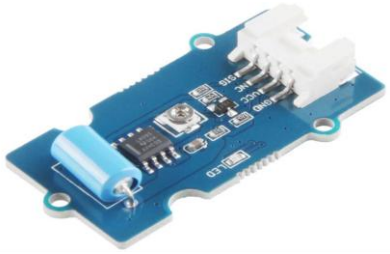


Figure 17: Vibration Sensor

The Grove - Vibration Sensor (SW-420) is a high sensitivity non-directional vibration sensor. When the module is stable, the circuit is turned on and the output is high. When the movement or vibration occurs, the circuit will be briefly disconnected and output low. At the same time, you can also adjust the sensitivity according to your own needs. All in all, this is a perfect module for vibration or tilt sensor.

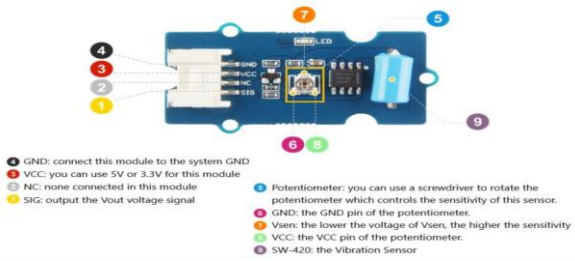
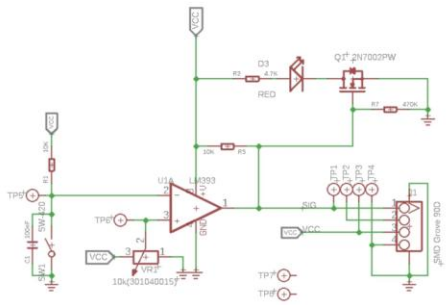


Figure 18: Pin Map



Figurev19: Schematic

First, let's begin with the SW1 which at the lower left corner. Actually, the SW1 is the vibration module SW-420. When the module is in a stable state, the module is turned on. Pin2 of U1A is connected to the GND though SW1.

The VR1 is the potentiometer, the Pin2 of the potentiometer is connected to the Pin3 of the U1A.

The U1A is a comparators. For the comparators,

Vout={High,if V+>V−Low,if V+<V−Vout={High,if V+>V−Low,if V+<V−

V+ connects to Pin3, V- connects to Pin2, Vout connects to Pin1.

For the V+ you can adjust it by rotate the potentiometer, for instance, we can make it VCC/2VCC/2.

For the V-, it depends on the SW1(SW-420):

• If this module is in a stable state, the SW1 is turned on, Pin2 of U1A is connected to the GND through SW1. It will be:

V−=0VV+=VCC/2}Vout=HighV−=0VV+=VCC/2}Vout=High

If the module vibrates or tilts, the SW1 will be turned off, the voltage of V- will be pulled up by the VCC through R1. Once the V- is higher than the VCC/2, then:

V−>VCC/2V+=VCC/2}Vout=LowV−>VCC/2V+=VCC/2}Vout=Low

Now you can set the V+ to adjust the sensitivity, just remember: the lower the voltage of V+, the higher the sensitivity

Buzzer

Figure 20: Buzzer

A buzzer or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

Buzzer Pin Configuration

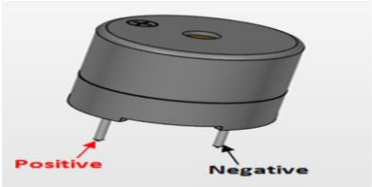


Figure 21: Buzzer Pin Configuration

IV. IMPLEMENTATION

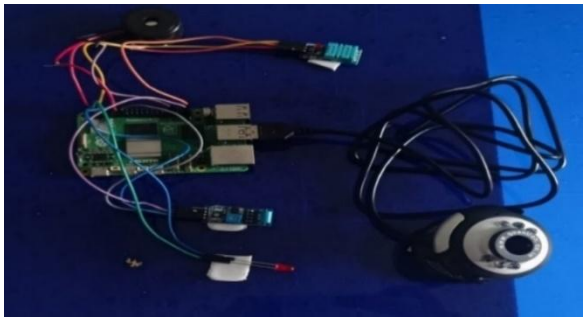


Figure 22: hardware setup

Step 1: Powering Up Raspberry Pi 5

Insert the MicroSD card (pre-installed with Raspberry Pi OS) into the slot.

Connect the power adapter to the Raspberry Pi's USB-C power port.

Wait for the boot process to complete. The LED on the board should turn on, indicating that the Pi is powered.

Step 2: Connecting Raspberry Pi to Laptop

Use an HDMI cable to connect Raspberry Pi to an external monitor (or use SSH if headless).

Connect a USB keyboard and mouse to the Raspberry Pi.

Power on the laptop and ensure both the Raspberry Pi and laptop are on the same network.

If using SSH (headless setup):

Find the Raspberry Pi's IP address using:

```
bash

ip a
```

Connect to it using your laptop's terminal:

```
bash

ssh pi@<Raspberry_Pi_IP>
```

Default password: raspberry

Step 3: Opening the Terminal and Running Code

After logging in to Raspberry Pi:

Open the Terminal from the menu or use Ctrl + Alt + T.

Navigate to your project folder:

```
bash

cd /home/pi/project-folder/
```

Open your Python code using a text editor:

```
bash

nano monitoring.py
```

If the code is already written, run it using:

```
bash

python3 monitoring.py
```

Step 4: Understanding Code Execution and Conditions

The monitoring.py script will check for conditions and act accordingly:

```
bash

echo -e "AT+CMGF=1\r" > /dev/ttyS0
echo -e 'AT+CMGS="+91XXXXXXXXXX"\r' > /dev/ttyS0
echo -e "Emergency Alert! Unusual condition detected.\x1A" > /dev/ttyS0
```

This sends an SMS alert in case of an emergency.

Result

Step-by-Step Execution of the System

Step 1: Continuous Monitoring

The camera is always active, checking for abnormal movements (e.g., sudden falls or stillness for a long time).

The DHT11 temperature sensor monitors room temperature.

The motion sensor detects any unusual inactivity or jerky movements.

Step 2: Triggering Alerts (Buzzer, LED, SMS, and Email)

Condition 1: Patient falls or abnormal movement detected

The camera captures an image immediately.

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

The buzzer and LED turn on as an emergency alert.

The system sends an SMS and email with the captured image.

Condition 2: High body temperature detected (>38°C)

If Temperature > 38°C → Alert is triggered

If Unusual Movement is Detected → Camera captures image

If Emergency Condition is Met → Buzzer sounds and SMS/email is sent

Step 5: Checking the GPRS Alert System

To send an alert via SMS when an abnormal condition is detected:

Ensure the GPRS module (SIM800L/SIM900A) is connected.

Run the following command to check if it's detected:

```
bash

ls /dev/tty*
```

Send an SMS using AT commands:

The buzzer turns on for 5 seconds.

A temperature warning message is sent via SMS and email.

Step 3: Sending SMS Alerts via GPRS Module

The SIM800L/SIM900A module is used to send SMS alerts to caregivers.

```
[ALERT] Emergency Detected!
Patient might need help.
Location: Home Care Unit
Check the monitoring camera feed.
- AI Monitoring System
```

Expected Real-Time Performance

How fast does the system respond?

Image Capture: Instant (0.5 - 1 sec)

Email Alert Sent: Within 3 - 5 seconds

SMS Sent via GPRS: Within 2 - 4 seconds

Buzzer Alert: Immediate (0 sec delay)

V. CONCLUSION

This project introduces an AI-driven real-time monitoring system designed specifically for paralysis patients, integrating computer vision, IoT sensors, and Raspberry Pi 5 to enhance patient safety. The system continuously analyzes live data, detecting unusual movements or distress conditions, and instantly triggers alerts via emails, SMS notifications, buzzer alarms, and LED signals to caregivers. The implementation of AI-based anomaly detection ensures rapid response in emergency situations, reducing the risks associated with delayed assistance. This solution demonstrates how intelligent automation can improve healthcare accessibility and efficiency for individuals with mobility impairments. Future enhancements could focus on expanding cloud-based analytics, mobile application support, and adaptive AI models for personalized patient care. With further refinements, this system holds potential for scalable, real-world deployment, ultimately improving the quality of life for paralysis patients.

VI. FUTURE SCOPE

The AI-enabled smart monitoring system for paralysis patients has vast potential for future advancements. Enhancing machine learning models with real-time adaptation can improve anomaly detection accuracy, ensuring more precise identification of emergencies. Cloud integration can enable secure remote data access, allowing doctors and caregivers to monitor patients from anywhere. Further, incorporating wearable biosensors alongside the existing AI vision system can provide a more comprehensive health assessment, tracking vital signs such as heart rate and oxygen levels. Developing a mobile application with real-time notifications and video streaming would offer better caregiver accessibility and faster response times. Additionally, edge computing can be leveraged to process data locally on Raspberry Pi, reducing dependency on external servers and improving system efficiency. Expanding the system's functionality to support multi patient monitoring in hospitals and home-care setups can make it a scalable healthcare solution. Future research can focus on AI-driven predictive analytics to anticipate medical complications before they occur, further enhancing patient safety and care quality.

REFERENCES

[1] R. K. Gupta, S. K. Sinha, and A. Verma, "AI-Enabled Remote Patient Monitoring System Using IoT," in SmartTechCon 2018, 2018. Available: https://ieeexplore.ieee.org/document/8747345

[2] M. Patel and A. Shah, "Wireless Sensor Networks for Healthcare Monitoring: A Review," International Journal of Computer Applications, vol. 182, no. 39, pp. 22-29, 2019. Available: https://www.ijcaonline.org/archives/volume182/number39/31732-2019919216

[3] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," Journal of Electrical and Computer Engineering, vol. 2017, pp. 1-

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 3, pp 43-59, March-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.403006

25, 2017. Available: https://www.hindawi.com/journals/jece/2017/9324035/

[4] M. S. Hossain, "Cloud-Supported Cyber-Physical Telemonitoring System for Patient-Centric e-Health Care," IEEE Systems Journal, vol. 11, no. 1, pp. 118-127, 2017. Available: https://ieeexplore.ieee.org/document/7460651

[5] L. Catarinucci et al., "An IoT-Aware Architecture for Smart Healthcare Systems," IEEE Internet of Things Journal, vol. 2, no. 6, pp. 515-526, 2015. Available: https://ieeexplore.ieee.org/document/7306544

[6] H. Alemdar and C. Ersoy, "Wireless Sensor Networks for Healthcare: A Survey," Computer Networks, vol. 54, no. 15, pp. 2688-2710, 2010. Available: https://www.sciencedirect.com/science/article/pii/S1389128610001079

[7] S. M. Riazul Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for Health Care: A Comprehensive Survey," IEEE Access, vol. 3, pp. 678-708, 2015. Available: https://ieeexplore.ieee.org/document/7113786

[8] M. Patel and A. Shah, "Wireless Sensor Networks for Healthcare Monitoring: A Review," International Journal of Computer Applications, vol. 182, no. 39, pp. 22-29, 2019. Available: https://www.ijcaonline.org/archives/volume182/number39/31732-2019919216

[9] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," Journal of Electrical and Computer Engineering, vol. 2017, pp. 1-25, 2017. Available: https://www.hindawi.com/journals/jece/2017/9324035/

[10] M. S. Hossain, "Cloud-Supported Cyber-Physical Telemonitoring System for Patient-Centric e-Health Care," IEEE Systems Journal, vol. 11, no. 1, pp. 118-127, 2017. Available: https://ieeexplore.ieee.org/document/7460651

[11] L. Catarinucci et al., "An IoT-Aware Architecture for Smart Healthcare Systems," IEEE Internet of Things Journal, vol. 2, no. 6, pp. 515-526, 2015. Available: https://ieeexplore.ieee.org/document/7306544

[12] H. Alemdar and C. Ersoy, "Wireless Sensor Networks for Healthcare: A Survey," Computer Networks, vol. 54, no. 15, pp. 2688-2710, 2010. Available: https://www.sciencedirect.com/science/article/pii/S1389128610001079

[13] S. M. Riazul Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for Health Care: A Comprehensive Survey," IEEE Access, vol. 3, pp. 678-708, 2015. Available: https://ieeexplore.ieee.org/document/7113786

Citation of this Article:

G.Yeswanthi, I. Rajshekar, S.Vyshnavi, P.Snehalatha, P.Thanuja, & T.M.Vasim Akram. (2025). AI-Enabled Smart Monitoring System for Paralysis Patient Using Raspberry Pi-5. *International Current Journal of Engineering and Science - ICJES*, 4(3), 43-59. Article DOI: https://doi.org/10.47001/ICJES/2025.403006

****

