A minimal OS for the Raspberry Pi, suitable for educational purposes

Team Code Conqueror

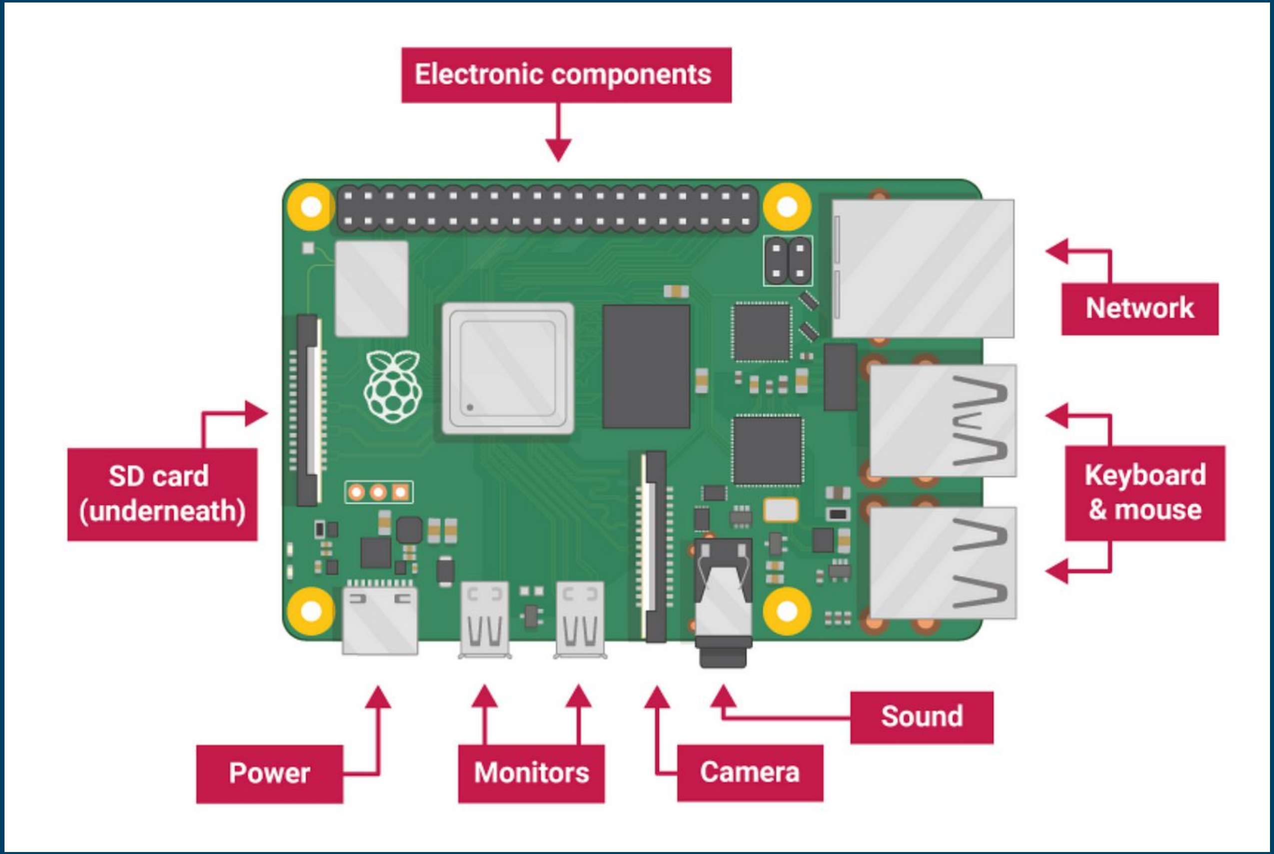
What is Raspberry Pi?

•The Raspberry Pi is a series of compact single-board computers created by the Raspberry Pi Foundation in the United Kingdom to support the teaching of basic computer science in schools and underdeveloped nations.

• The initial model was significantly more successful than expected, selling outside of its intended market for applications such as robotics.

• Before February 2015, almost 5 million Raspberry Pis had been sold, making it the best-selling British computer. They have sold 11 million devices by November 2016.

Raspberry Pi board



Components


GPIO
DSI LCD with connector
CSI camera with connector
Kingston
micro SD
ADAPTER
SD
MicroSD
3.5mm audio jack
USB 2.0
Micro USB
HDMI connector
Ethernet connector
RCA video connector

Components

- **Essential:**

→Raspberry Pi board

→Prepared Operating System SD Card

→USB keyboard

→Display (with HDMI, DVI, or Composite input)

→Power Supply

- **Highly suggested extras include:**

→USB mouse

→Internet connectivity - LAN cable

→Powered USB Hub

→Case

Programming language

- The Raspberry Pi Foundation recommends Python
- Any language which will compile for ARMv6 can be used
- Installed by default on the Raspberry Pi:

→C

→C++

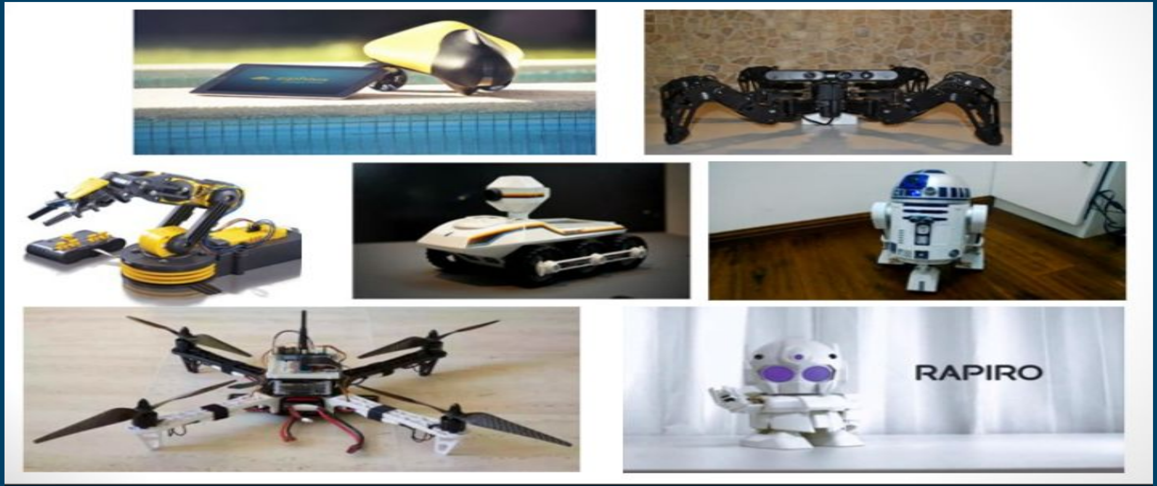
→Java

→Scratch

→Ruby

Robots and drones

Drones: A Raspberry Pi can give a drone significant computing power on board, allowing access to technologies such as AI. For example, a camera-equipped Raspberry Pi can use computer vision to determine where a drone is when flying indoors.



Robotics:A Raspberry Pi can be used for a wide range of robotics applications, including desktop robots, mobile robots, industrial robots, and more. For example, a Raspberry Pi can be used to construct a six-axis collaborative robot arm.

Timeline Of Raspberry Pi ->

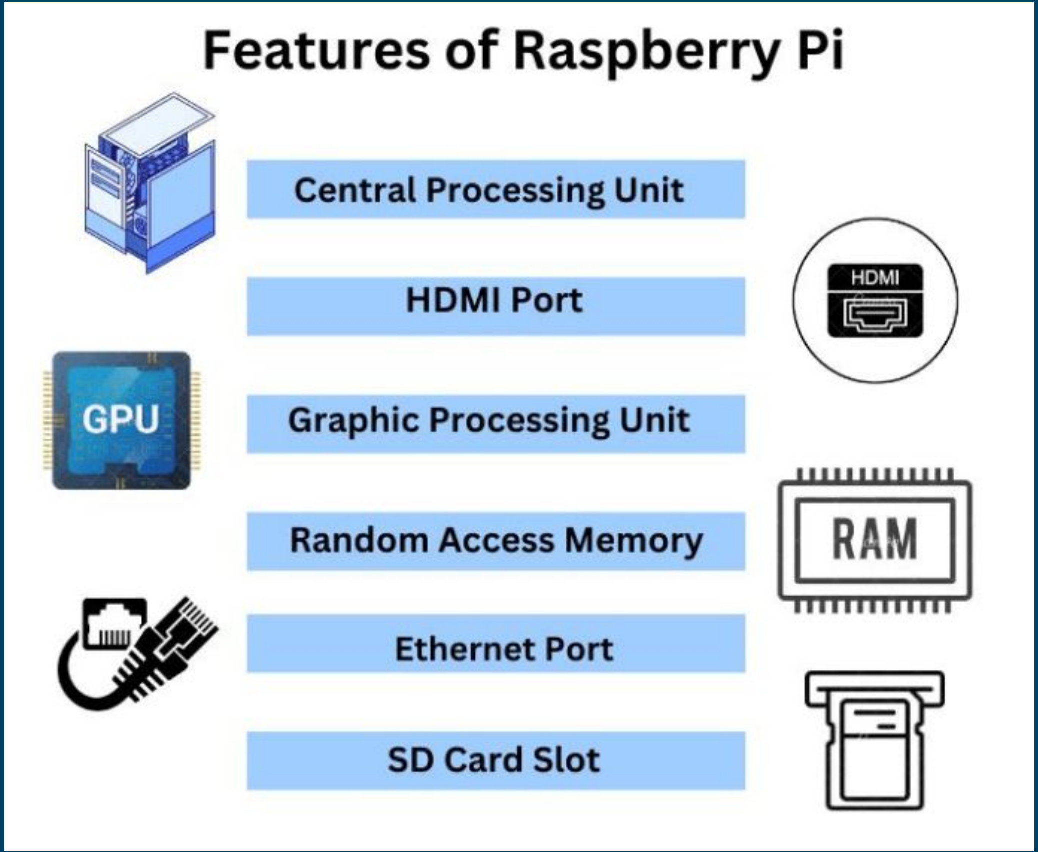
- In February 2012, the first generation (Raspberry Pi 1 Model B) was released. Model A, a simpler and less expensive model, followed.

- In 2014, the Raspberry Pi Foundation released a board with an updated design called the Raspberry Pi 1 Model B+. These boards are roughly the size of a credit card and represent the basic mainline form-factor.

- A year later, improved A+ and B+ models were released. In April 2014, a "compute module" for embedded applications was published, while in November 2015, a Raspberry Pi Zero with lower size and decreased input/output (I/O) and general-purpose input/output (GPIO) capabilities was released for US$5.

- In February 2015, the Raspberry Pi 2 with additional RAM was released.
- The Raspberry Pi 3 Model B, which was released in February 2016, includes on-board WiFi, Bluetooth, and USB boot capabilities.
- As of January 2017, the newest mainline Raspberry Pi is the Raspberry Pi 3 Model B.
- Raspberry Pi boards range in price from $5 to $35 USD.
- On February 28, 2017, the Raspberry Pi Zero W, which is identical to the Raspberry Pi Zero but features the Wi-Fi and Bluetooth functionality of the Raspberry Pi 3, was released for US$10.

Features


Features of Raspberry Pi
Central Processing Unit
HDMI Port
HDMI
Graphic Processing Unit
GPU
Random Access Memory
RAM
Ethernet Port
SD Card Slot

Features

- All models use a Broadcom system on a chip (SoC), which features an ARM-compatible CPU and an on-board graphics processing unit (GPU, a VideoCore IV).

- The Pi 3's CPU speed goes from 700 MHz to 1.2 GHz, and on-board memory spans from 256 MB to 1 GB RAM.

- Secure Digital (SD) cards in SDHC or MicroSDHC sizes are utilised to store the operating system and programme memory.

- Most boards offer one to four USB ports, HDMI and composite video output, and an audio 3.5 mm phono connector.

- A number of GPIO pins provide lower level output and support standard protocols such as I2C.

- The B-models include an 8P8C Ethernet port, whereas the Pi 3 and Pi Zero W include built-in Wi-Fi 802.11n and Bluetooth.

Key component

- CPU: The Raspberry Pi has a single-core ARMv6 700 MHz processor. The Raspberry Pi 4 has a quad-core Cortex-A72 64-bit CPU running at 1.5GHz.
- GPU:The Raspberry Pi has a VideoCore IV GPU. The Raspberry Pi 4 has a Broadcom VideoCore VI GPU running at 0.5GHz.
- RAM:The Raspberry Pi 1 has 512MB of RAM. The Raspberry Pi 4 has 40 GPIO pins.
- Storage:The Raspberry Pi uses an SD card for its operating system and data storage. The Raspberry Pi 400 has 4GB of RAM.
- GPIO pins:The Raspberry Pi has a row of GPIO (general-purpose input/output) pins along the top edge of the board. The Raspberry Pi 1 Models A and B have only the first 26 pins.

Variations of Raspberry Pi

Hardware platform:

- Raspberry Pi Zero ($5) :Cost-effective, basic model
- Standard Raspberry Pi:Standard model for general-purpose computing
- Raspberry Pi 2:Improved performance and features
- Raspberry Pi 3 (with Wifi + Bluetooth):Enhanced connectivity opt

Software platform:

- Noobs: Beginner-friendly operating system installer
- Raspbian:Official operating system based on Debian, optimized for Raspberry Pi
- 3rd OS :Customizable, depending on project needs

History

The Foundation's intention was to sell two versions, each priced at $25 and $35 USD.

They began accepting orders for the more expensive Model B on February 29, 2012, and the less expensive Model A on February 4, 2013. and the even more affordable (US$20) A+ on November 10, 2014.

On November 26, 2015, the Raspberry Pi Zero, the cheapest Raspberry Pi yet, was released for $5 .

Why to use raspberry Pi

- The Raspberry Pi is a good choice for people new to computing and programming because it's simple, cheap, and runs on Linux.

- The Raspberry Pi 4 Model B costs $35.
- The 2GB Raspberry Pi 4 costs $45.
- The 8GB Raspberry Pi 4B costs $75.
- The Raspberry Pi 3 – Model A+ costs $25

- The cheapest Raspberry Pi is the Raspberry Pi Zero, which costs $5.

Raspberry Pi - Operating System:

- A Raspberry Pi operating system (OS) is a specialized software designed to run on Raspberry Pi, a small, affordable, single-board computer. The Raspberry Pi OS is responsible for managing the computer's hardware resources and providing a user interface for interaction.

Here are the core functions of a Raspberry Pi operating system:

Process Management:

- This involves managing the execution of programs or processes. The OS is responsible for starting, pausing, terminating, and scheduling processes to make the most efficient use of the CPU.

Memory Management:

- The OS allocates and deallocates memory for processes. It ensures that each program gets the required memory space and protects them from interfering with each other.

Device Management:

- This involves managing input and output devices such as USB peripherals, displays, keyboards, and networking interfaces. The OS facilitates communication between software and hardware components.

User Interface (UI):

- The UI provides a way for users to interact with the Raspberry Pi. It can be a command-line interface (CLI) where users type commands, or a graphical user interface (GUI) with windows, icons, buttons, and menus.

Networking:

- The OS handles network connections, allowing the Raspberry Pi to communicate with other devices over a network, be it wired or wireless.
- This includes configuring IP addresses, managing protocols (like TCP/IP), and handling network services.

Security:

- The OS implements security measures to protect the system from unauthorized access, viruses, and other threats. This can include user authentication, encryption, and firewall settings.

File System:

- The file system manages how data is stored, organized, and accessed on the storage media (like SD cards or USB drives). It provides a hierarchical structure for files and directories.

Driver Management:

- Drivers are software components that allow the OS to communicate with specific hardware devices. The OS loads and manages these drivers to ensure proper functioning of attached peripherals.

Updates and Maintenance :

- The OS may include mechanisms for updating itself and installed software packages to patch security vulnerabilities or add new features.

File System Permissions:

- The OS enforces access controls on files and directories, ensuring that only authorized users or processes can read, write, or execute them.

Error Handling:

- The OS is responsible for detecting and managing errors that may occur during operation. This can include hardware errors, software crashes, or other unexpected events.

Summary

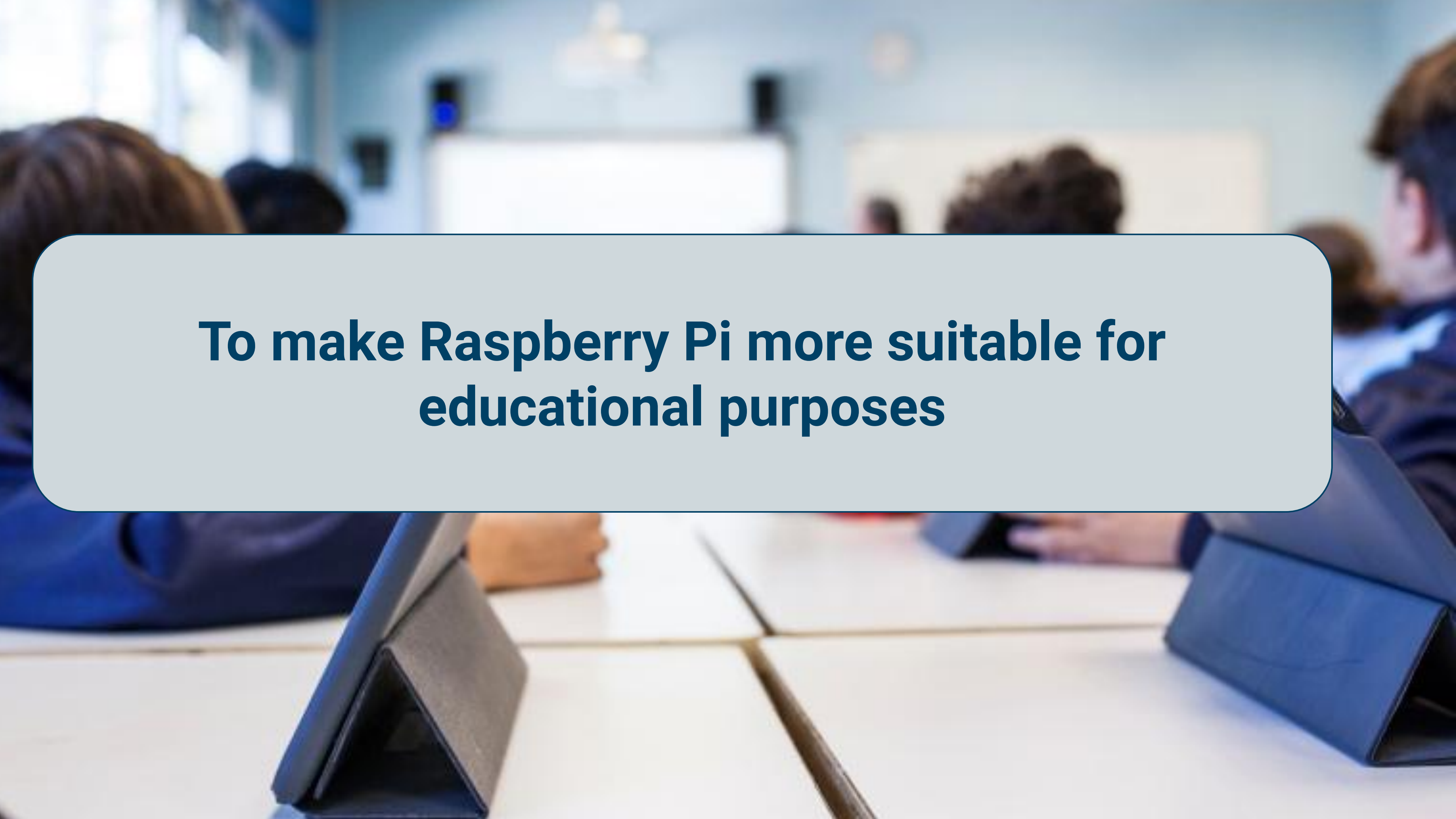
The Raspberry Pi is a low-cost, multifunctional single-board computer used in education and home improvement projects. It is compatible with Linux-based operating systems and includes GPIO pins for hardware interface. It's a popular choice for learning, prototyping, and creative endeavours.

PROS

- Affordable
- Versatile for a wide range of projects
- Strong community and support
- Low power consumption
- Compact size
- Rich GPIO support
- Customizable
- Open-source
- Great for education.

Cons

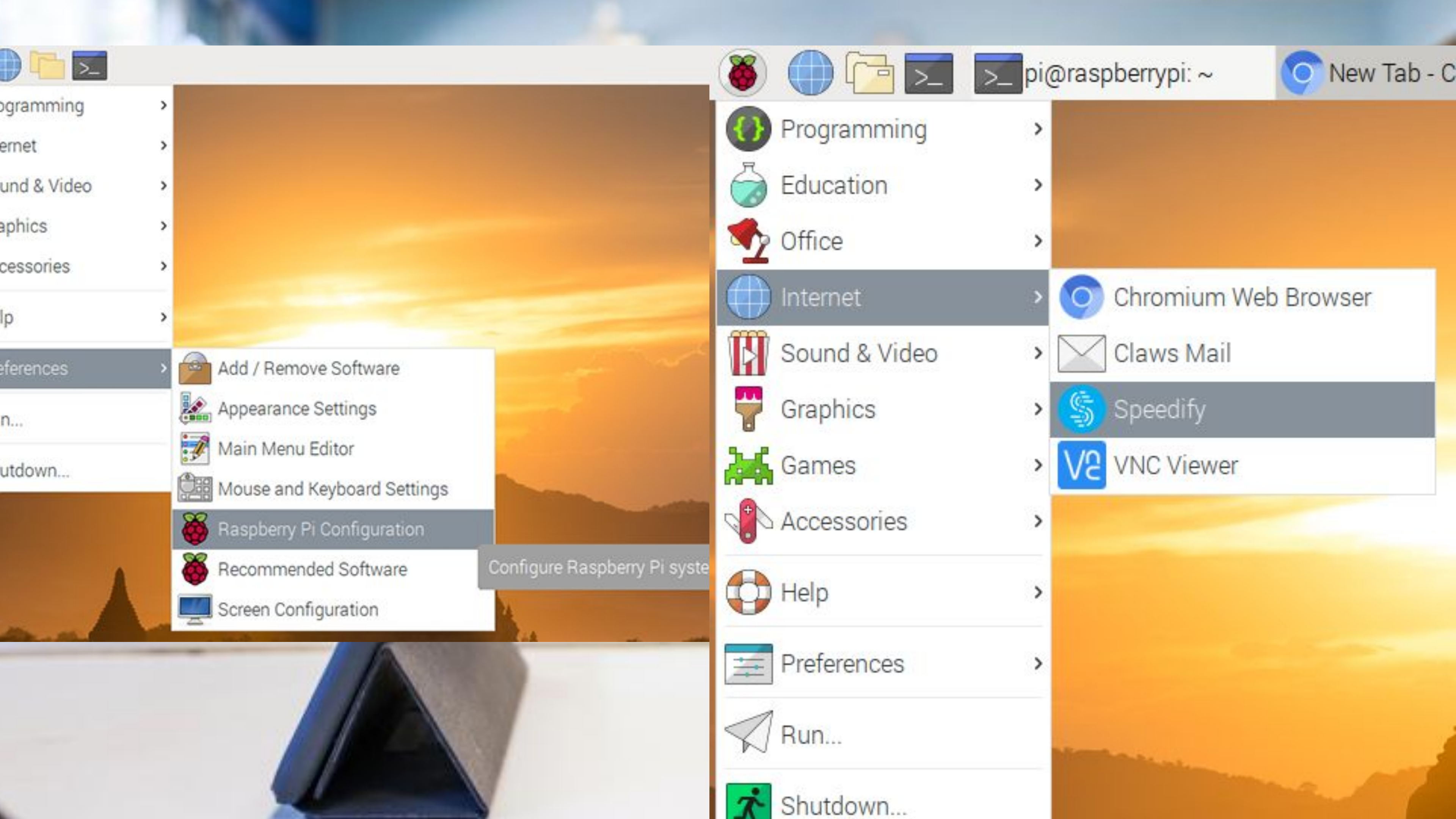
- Limited performance.
- Limited storage options.
- Noisy audio output.
- Limited graphics capabilities.
- Compatibility issues.
- No built-in battery backup.
- Learning curve for beginners.
- Limited networking options on some models



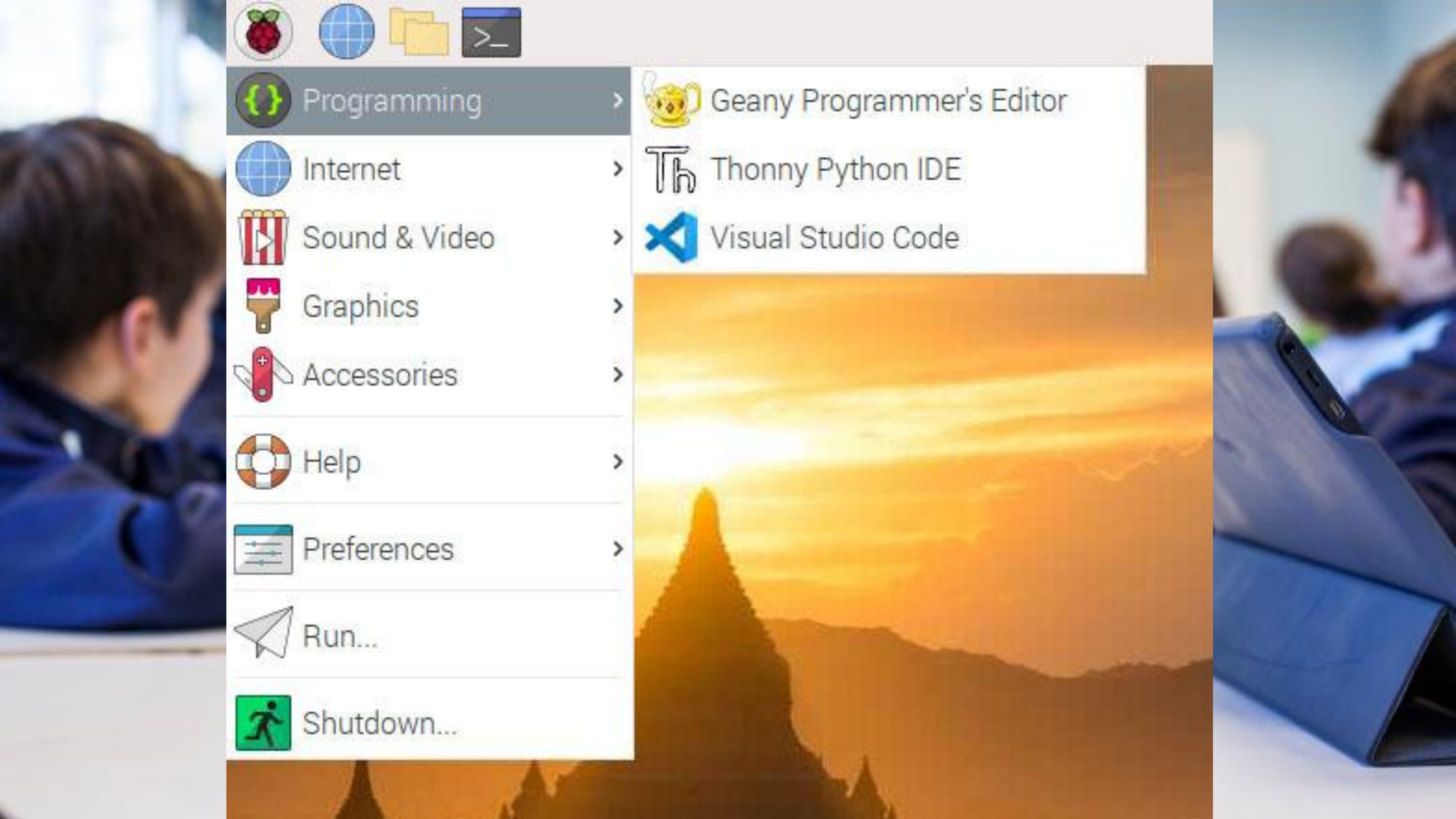
To make Raspberry Pi more suitable for educational purposes

<u>Raspberry Pi: A Game-Changer for Education</u>

This compact device offers students hands-on experience in programming, electronics, and robotics, making it an invaluable resource for schools and educational institutions.



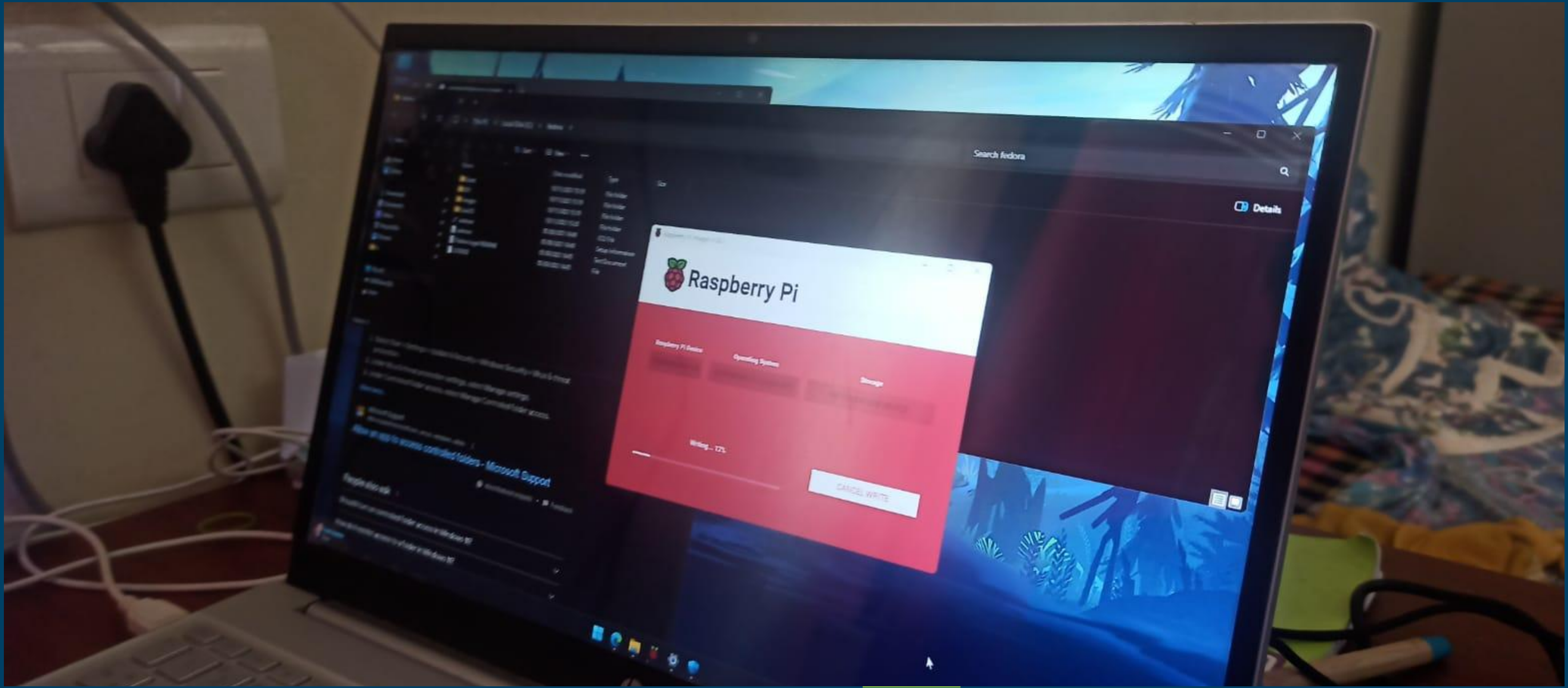
ogramming
ernet
und & Video
aphics
cessories
lp
eferences
n...
utdown...
Add / Remove Software
Appearance Settings
Main Menu Editor
Mouse and Keyboard Settings
Raspberry Pi Configuration
Recommended Software
Screen Configuration
Configure Raspberry Pi syste
pi@raspberrypi: ~
New Tab - C
Programming
Education
Office
Internet
Sound & Video
Graphics
Games
Accessories
Help
Preferences
Run...
Shutdown...
Chromium Web Browser
Claws Mail
Speedify
VNC Viewer


Programming
Internet
Sound & Video
Graphics
Accessories
Help
Preferences
Run...
Shutdown...
Geany Programmer's Editor
Thonny Python IDE
Visual Studio Code

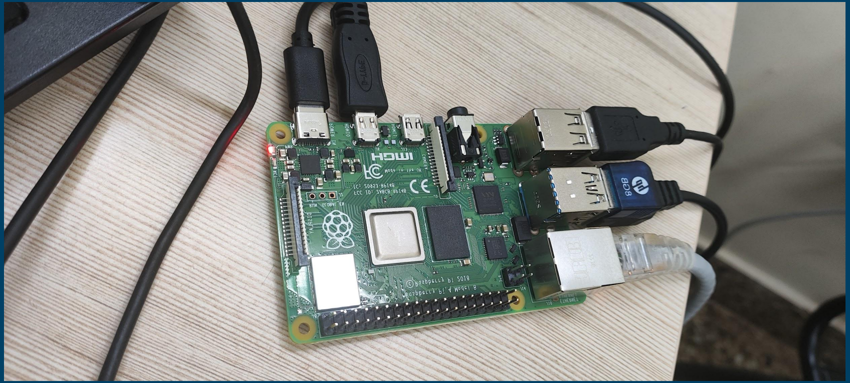

(7 customer reviews)
Availability: In stock
SKU: 76167
Add to Wishlist
1. LCD Type: TFT.
2. Touch Screen Type: Resistive Screen.
3. Backlight: Yes, LED.
4. Resolution (Pixel): 320 × 480.
5. Aspect Ratio: 8:5.
6. Touch Controller: XPT2046.
₹ 1,127.00
MINI
TOUCH
LCD

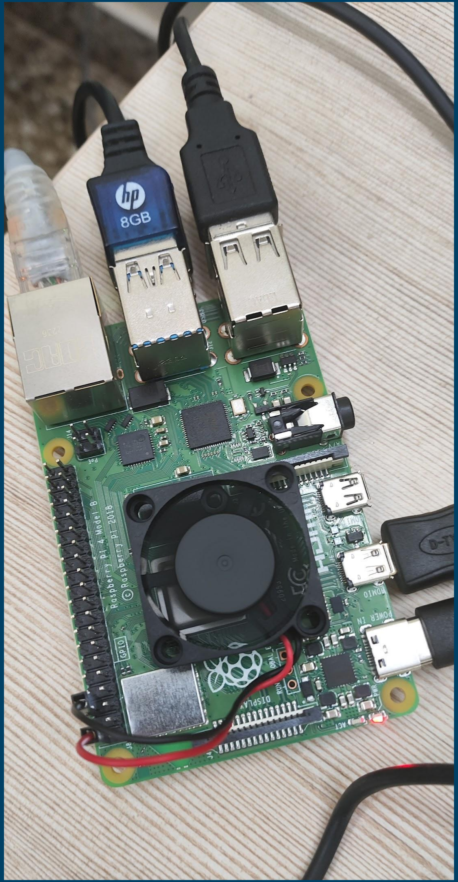
How We Boot this Raspberry pi

1. Insert a bootable USB/sd card that has the Raspberry Pi OS.
2. Connect any necessary accessories (keyboard, mouse, monitor, and power).
3. Switch on the Raspberry Pi.It initialises and loads the operating system from the USB Drive.
4. You can access it by logging in.
5. It can also boot into a custom application or script for your project.


Search fedora
Details
Raspberry Pi

Set up of raspberry pi





- **<u>Bootloader to load your kernel.</u>**

Between the two USB drives, one exhibits superior read and write operations, resulting in faster boot times compared to the other.Link: https://drive.google.com/file/d/1IQwFM1Q5INQUkR4yqIEBrxs7tAKA6IHT/view?usp=sharing

<u>TESTED WITH TWO USB DRIVES</u>

- **<u>We see difference in opening and rebooting time with raspberry pi</u>**

<u>So this is a major factor try to use a Good read and write speed, SD card / USB Drive</u>

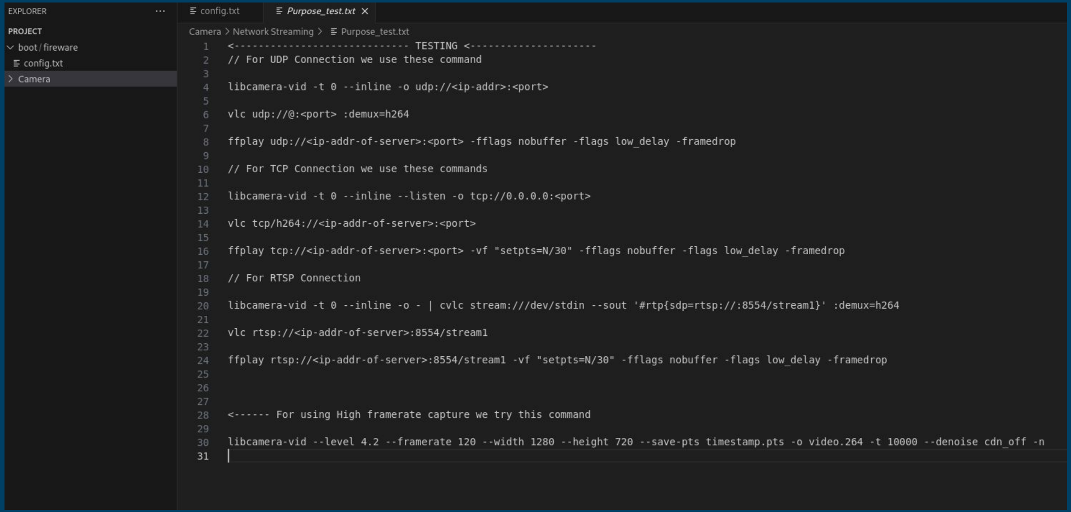
Raspberry Pi Cameras

```
File Name                       : test.dng
Directory                       : .
File Size                       : 24 MB
File Modification Date/Time     : 2021:08:17 16:36:18+01:00
File Access Date/Time           : 2021:08:17 16:36:18+01:00
File Inode Change Date/Time     : 2021:08:17 16:36:18+01:00
File Permissions                : rw-r--r--
File Type                       : DNG
File Type Extension             : dng
MIME Type                       : image/x-adobe-dng
Exif Byte Order                 : Little-endian (Intel, II)
Make                            : Raspberry Pi
Camera Model Name               : /base/soc/i2c0mux/i2c@1/imx477@1a
Orientation                     : Horizontal (normal)
Software                        : libcamera-still
Subfile Type                    : Full-resolution Image
Image Width                     : 4056
Image Height                    : 3040
Bits Per Sample                 : 16
Compression                     : Uncompressed
Photometric Interpretation      : Color Filter Array
Samples Per Pixel               : 1
Planar Configuration            : Chunky
CFA Repeat Pattern Dim          : 2 2
CFA Pattern 2                   : 2 1 1 0
Black Level Repeat Dim          : 2 2
Black Level                     : 256 256 256 256
White Level                     : 4095
DNG Version                     : 1.1.0.0
DNG Backward Version            : 1.0.0.0
Unique Camera Model             : /base/soc/i2c0mux/i2c@1/imx477@1a
Color Matrix 1                  : 0.8545269369 -0.2382823821 -0.09044229197 -0.1890484985 1.063961506 0.10
As Shot Neutral                 : 0.4754476844 1 0.413686484
Calibration Illuminant 1        : D65
Strip Offsets                   : 0
Strip Byte Counts               : 0
Exposure Time                   : 1/20
ISO                             : 400
CFA Pattern                     : [Blue,Green][Green,Red]
Image Size                      : 4056x3040
Megapixels                      : 12.3
Shutter Speed                   : 1/20
```

libcamera and libcamera-apps

- **So the point is, students need webcam for classes and all**

Raspberry Pi Camera



libcamera and libcamera-apps

- **So the point is, students need webcam for classes and all**
- **Understand and change in Network Streaming**

<------Like For using High frame rate capture we try this command —->

libcamera-vid --level 4.2 --framerate 120 --width 1280 --height 720 --save-pts timestamp.pts -o video.264 -t 10000 --denoise cdn_off -n

Raspberry Pi Camera

--sharpness Set image sharpness <number>

--shutter Set the exposure time in microseconds <number>

--ev Set EV compensation <number>

--saturation Set image colour saturation <number>

--contrast Set image contrast <number>

--brightness Set image brightness <number>

- By Importing these things in our camera module file we can direct use these commands for changing in camera setting for our educational purpose

Raspberry Pi Reduce File Sizes

We don't have much space on the Raspberry Pi, which is why we modified the code to reduce the size of all image formats, such as JPG, PNG, ~~etc.~~"

jpeg.cpp

```cpp

        // Make all the EXIF data, which includes the thumbnail.

        jpeg_mem_len_t thumb_len = 0; // stays zero if no thumbnail
        unsigned int exif_len;
        create_exif_data(mem, info, metadata, cam_model, options, exif_buffer, exif_len, thumb_buffer, thumb_len);

        // Make the full size JPEG (could probably be more efficient if we had
        // YUV422 or YUV420 planar format).


//   <--------------- CHNAGES IN CODE jpeg.copy
//                 for reduce file sizes and quality
//                because we dont have so much memory -------->

        // Set lower quality for the final JPEG image (adjust the value as needed)
        options->quality = 10;

         // Set smaller restart interval (adjust the value as needed)
        options->restart = 8;

// <------  by above two lines we manually adjust the quality ----->

        jpeg_mem_len_t jpeg_len;
        YUV_to_JPEG((uint8_t *)(mem[0].data()), info, info.width, info.height, options->quality, options->restart,
                    jpeg_buffer, jpeg_len);
        LOG(2, "JPEG size is " << jpeg_len);

        // Write everything out.

        fp = filename == "-" ? stdout : fopen(filename.c_str(), "w");
        if (!fp)
            throw std::runtime_error("failed to open file " + options->output);
```

How above code works ?

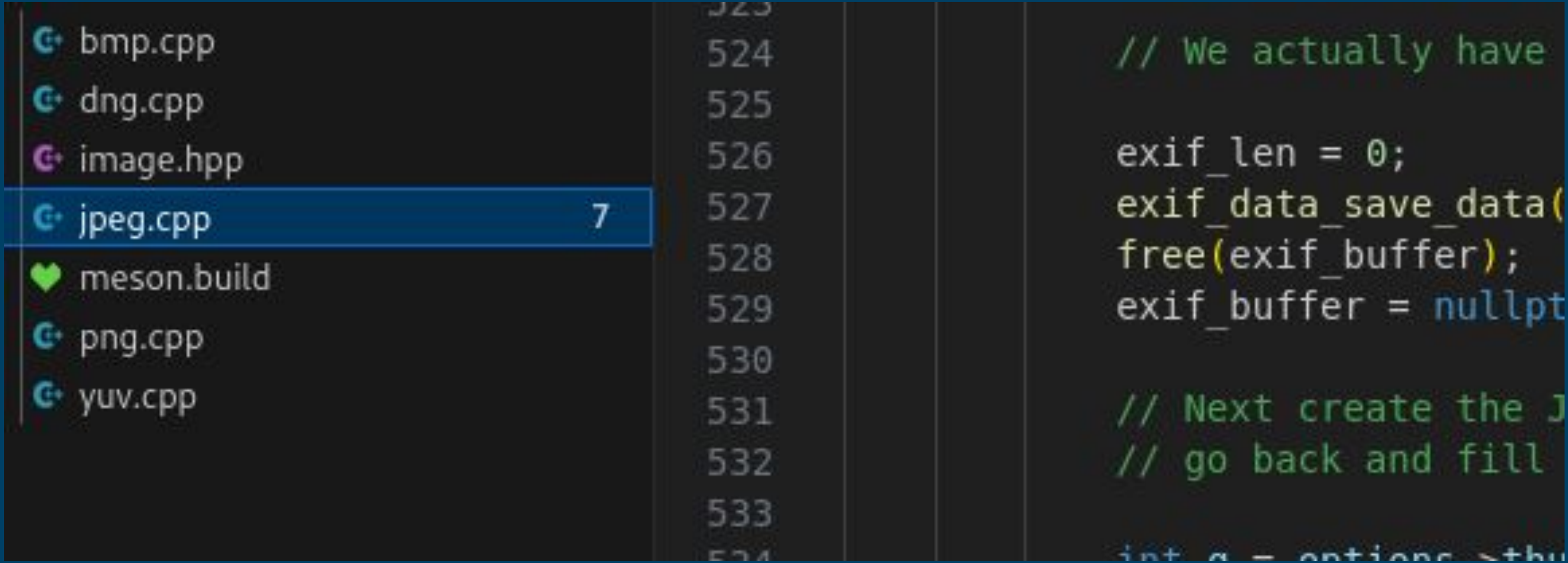
Lowering Quality for Higher Compression

```
options->quality = 10;  // Lower quality for higher compression
```

Smaller Restart Interval for Improved Compression:

```
options->restart = 8;  // Smaller restart interval may improve compression
```

Same changes we can do in other files formats



Next we optimize the encoder to work as a powerful enoder and reduce our file size more

mjpeg_encoder.cpp

```cpp
void MjpegEncoder::encodeJPEG(struct jpeg_compress_struct &cinfo, EncodeItem &item, uint8_t *&encoded_buffer,
                              size_t &buffer_len)
{
    // Copied from YUV420_to_JPEG_fast in jpeg.cpp.
    cinfo.image_width = item.info.width;
    cinfo.image_height = item.info.height;

    // 1. Lower JPEG Quality
    options_->quality = 5; // Adjust the quality as needed

    // Set default JPEG compression parameters in the cinfo structure.
    jpeg_set_defaults(&cinfo);

    // here cinfo in jpeg_compress_struct
    // Specify that raw data will be provided
    cinfo.raw_data_in = TRUE;

    // Set JPEG quality based on the adjusted value
    jpeg_set_quality(&cinfo, options_->quality, TRUE);

    // 2. Resize the image to a smaller resolution
    cinfo.image_width /= 2; // Adjust the new width as needed
    cinfo.image_height /= 2; // Adjust the new height as needed

    // 3. Adjust chroma subsampling ratio
    cinfo.comp_info[0].h_samp_factor = 2; // Adjust as needed
    cinfo.comp_info[0].v_samp_factor = 1; // Adjust as needed

    // 4. Enable progressive JPEG encoding
    jpeg_simple_progression(&cinfo);
    //  These above lines for adjust the horizontal and vertical chroma subsampling factors.

    // cinfo.input_components = 3;
    // cinfo.in_color_space = JCS_YCbCr;
    // cinfo.restart_interval = 0;

    // jpeg_set_defaults(&cinfo);
    // cinfo.raw_data_in = TRUE;
    // jpeg_set_quality(&cinfo, options_->quality, TRUE);
    encoded_buffer = nullptr;
    buffer_len = 0;
    jpeg_mem_len_t jpeg_mem_len;
    jpeg_mem_dest(&cinfo, &encoded_buffer, &jpeg_mem_len);
    jpeg_start_compress(&cinfo, TRUE);

    int stride2 = item.info.stride / 2;
    uint8_t *Y = (uint8_t *)item.mem;
    uint8_t *U = (uint8_t *)Y + item.info.stride * item.info.height;
```

How above code works ?

Above Four Changes works for these =>

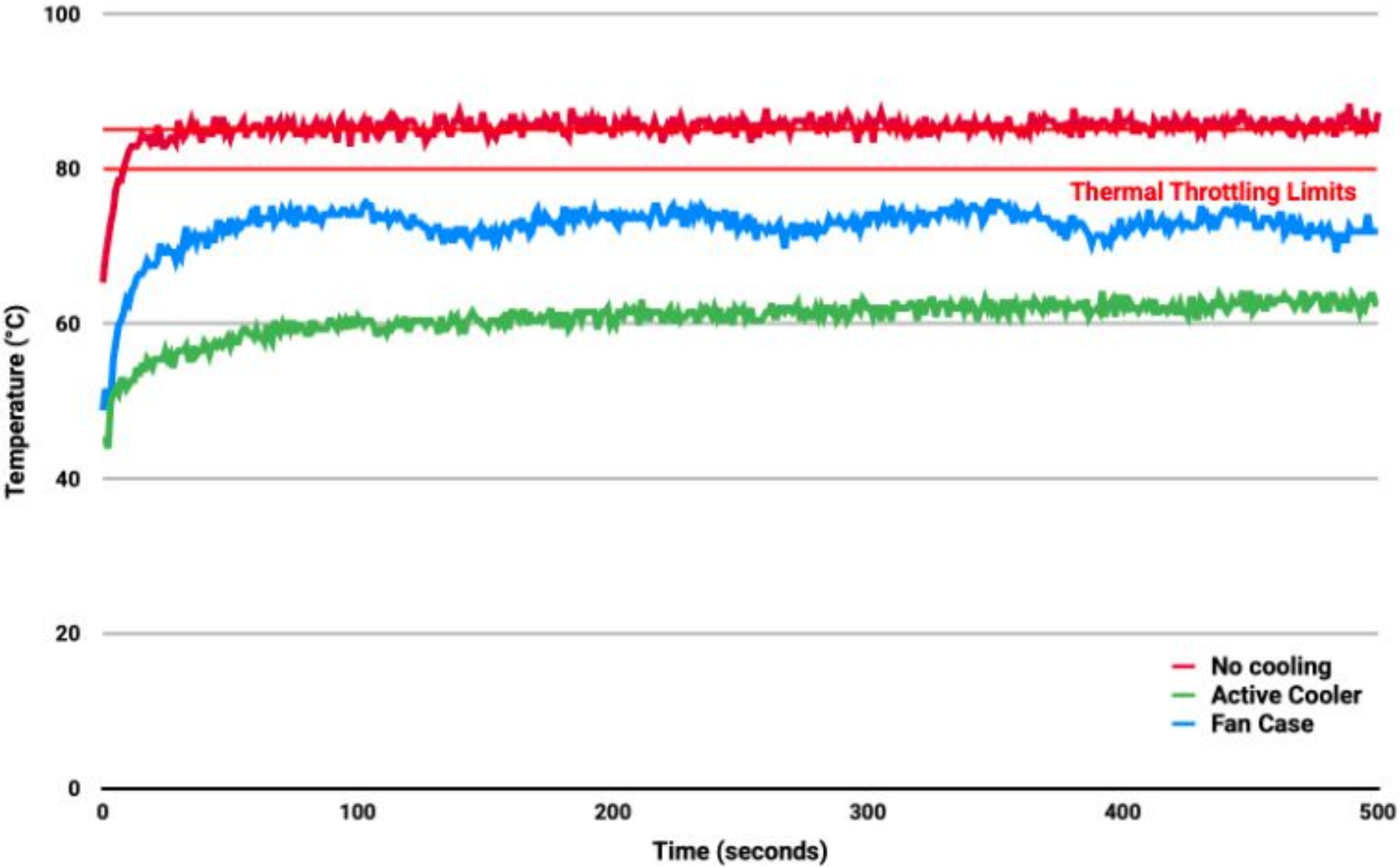
- Lower JPEG Quality
- Resize the Image to a Smaller Resolution
- Adjust Chroma Subsampling Ratio
- Enable Progressive JPEG Encoding

Same changes we can do in other files formats

```
┌──(gautamop㉿gautamop)-[~/Desktop/CS310/PROJECT/encoder]
└─$ ls
encoder.cpp       h264_encoder.hpp   meson.build          null_encoder.cpp
encoder.hpp       libav_encoder.cpp  mjpeg_encoder.cpp    null_encoder.hpp
h264_encoder.cpp  libav_encoder.hpp  mjpeg_encoder.hpp
```

Things we observed

After using the raspberry pi for a while the cpu becomes very warm.
For this we need to stop the processes which we don't need .



100
80
60
40
20
0
Temperature (°C)
Thermal Throttling Limits
0
100
200
300
400
500
Time (seconds)
No cooling
Active Cooler
Fan Case

Things we observed

vcgencmd pmic_read_adc

Bootloader - Shutdown Time

booloader > About_Chnages.txt

```
Default shutdown wattage is around 1 to 1.4W. However this can be decreased by manually editing the EEPROM
configuration,

sudo rpi-eeprom-config -e and change the settings to:



# Enable UART output during the boot process for debugging or information
BOOT_UART=1

# Power off the Raspberry Pi completely when halted (shutdown)
POWER_OFF_ON_HALT=1

# Define the boot order bit pattern
# Bit 0: USB mass storage device
# Bit 1: SD card
# Bit 2: Network boot (PXE)
# Bit 3: Boot from USB device
BOOT_ORDER=0xf416  # Binary: 1111 0100 0001 0110
# This means the Raspberry Pi will attempt to boot in the order of:
# 1. USB mass storage device
# 2. SD card
# 3. Network boot (PXE)
# 4. Boot from USB device
```

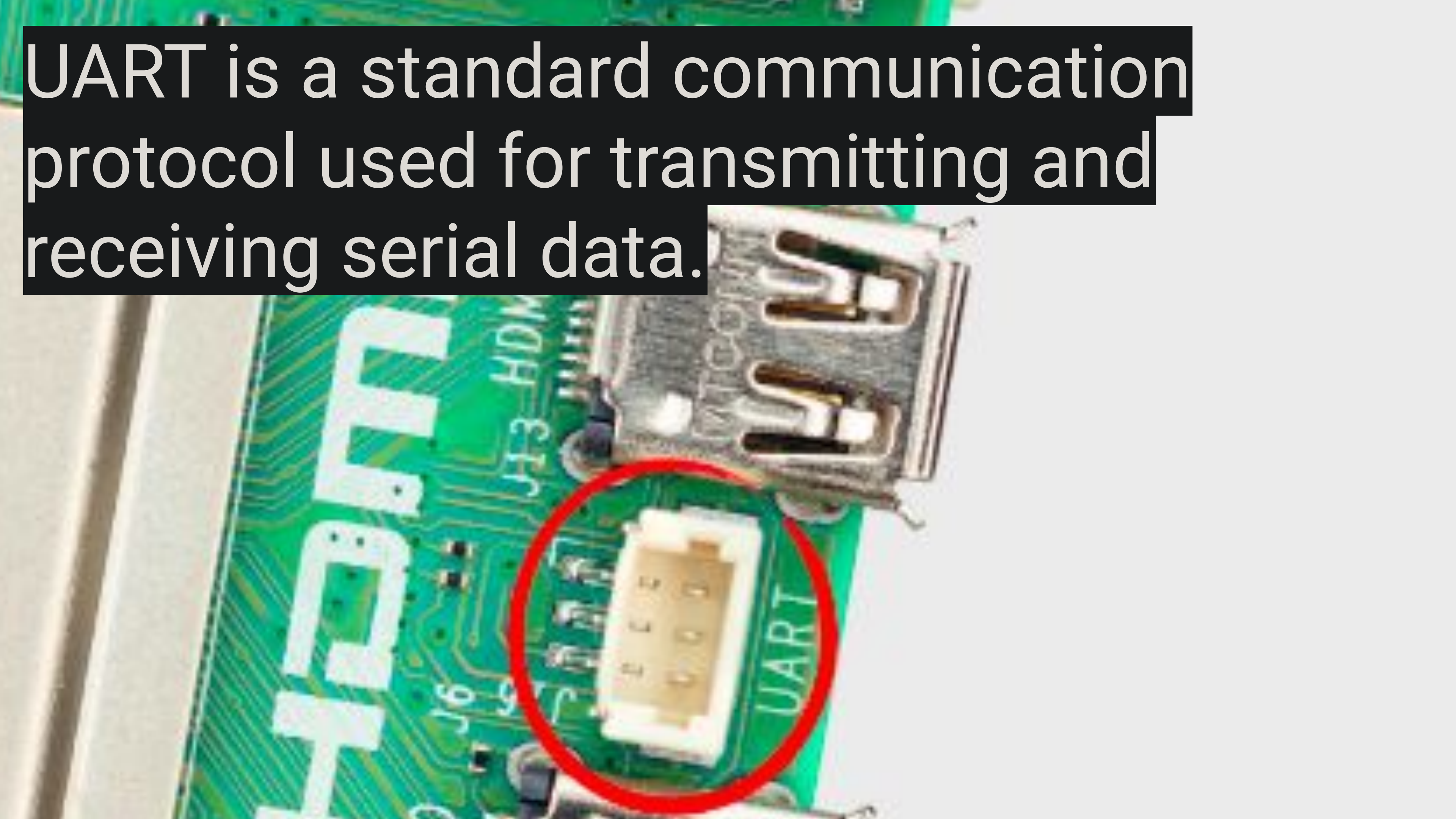
Bootloader - Shutdown Time

– enable Universal Asynchronous Receiver-Transmitter.
BOOT_UART=1

– Power Off the Raspberry Pi when halted (shutdown)
POWER_OFF_ON_HALT=1

– RP will attempt to boot in order
BOOT_ORDER=0xf416

UART is a standard communication protocol used for transmitting and receiving serial data.



Now we want to open some applications automatically when we start our Raspberry Pi

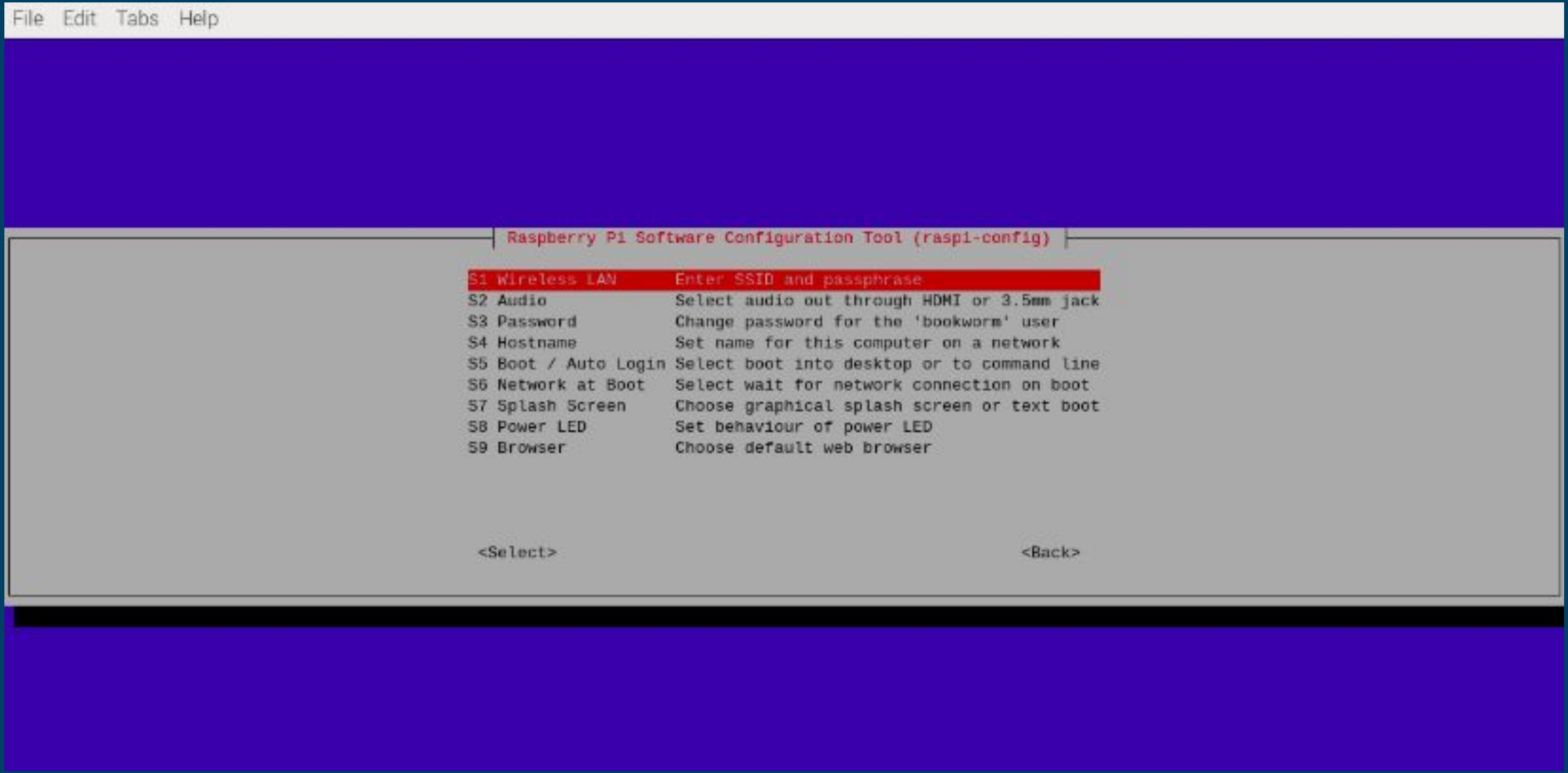
booloader > $ startup_script.sh

```bash
#!/bin/bash

# Enable Universal Asynchronous Receiver-Transmitter (UART)
BOOT_UART=1

# Power off the Raspberry Pi when halted (shutdown)
POWER_OFF_ON_HALT=1

# Raspberry Pi will attempt to boot in the specified order
BOOT_ORDER=0xf416

# Open Google Classroom in Chrome
chromium-browser https://classroom.google.com &

# You can add more lines for other applications or configurations

# End of script

```

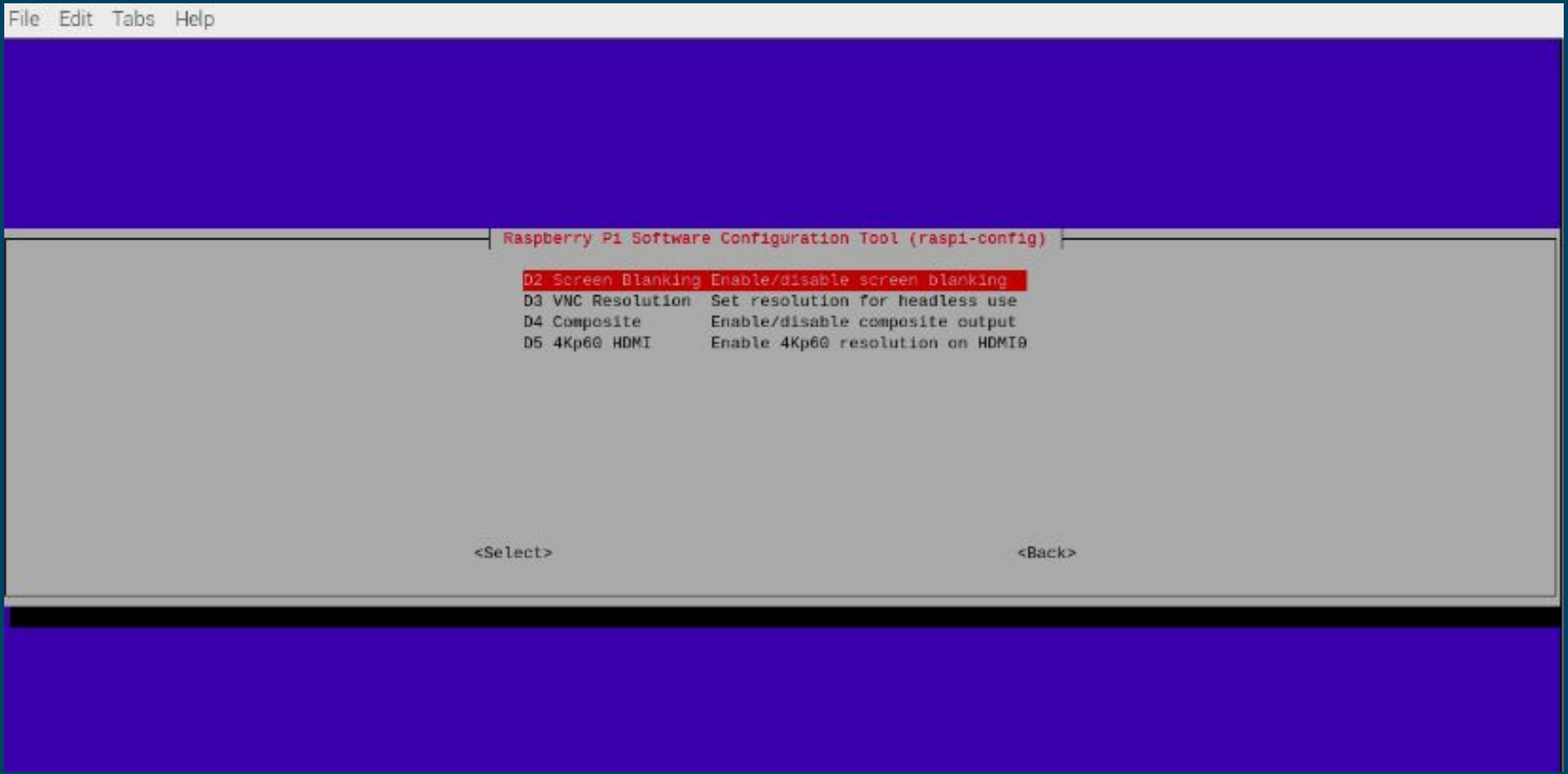
I am a student, and I don't want to open the classroom every time manually. So, I have attached my bash script with a bootloader. Now, when I open my Raspberry Pi, the script runs automatically and opens the classroom directly.

We can modify this according to our requirements. Additionally, we can add scripts for YouTube educational playlists, emails, etc

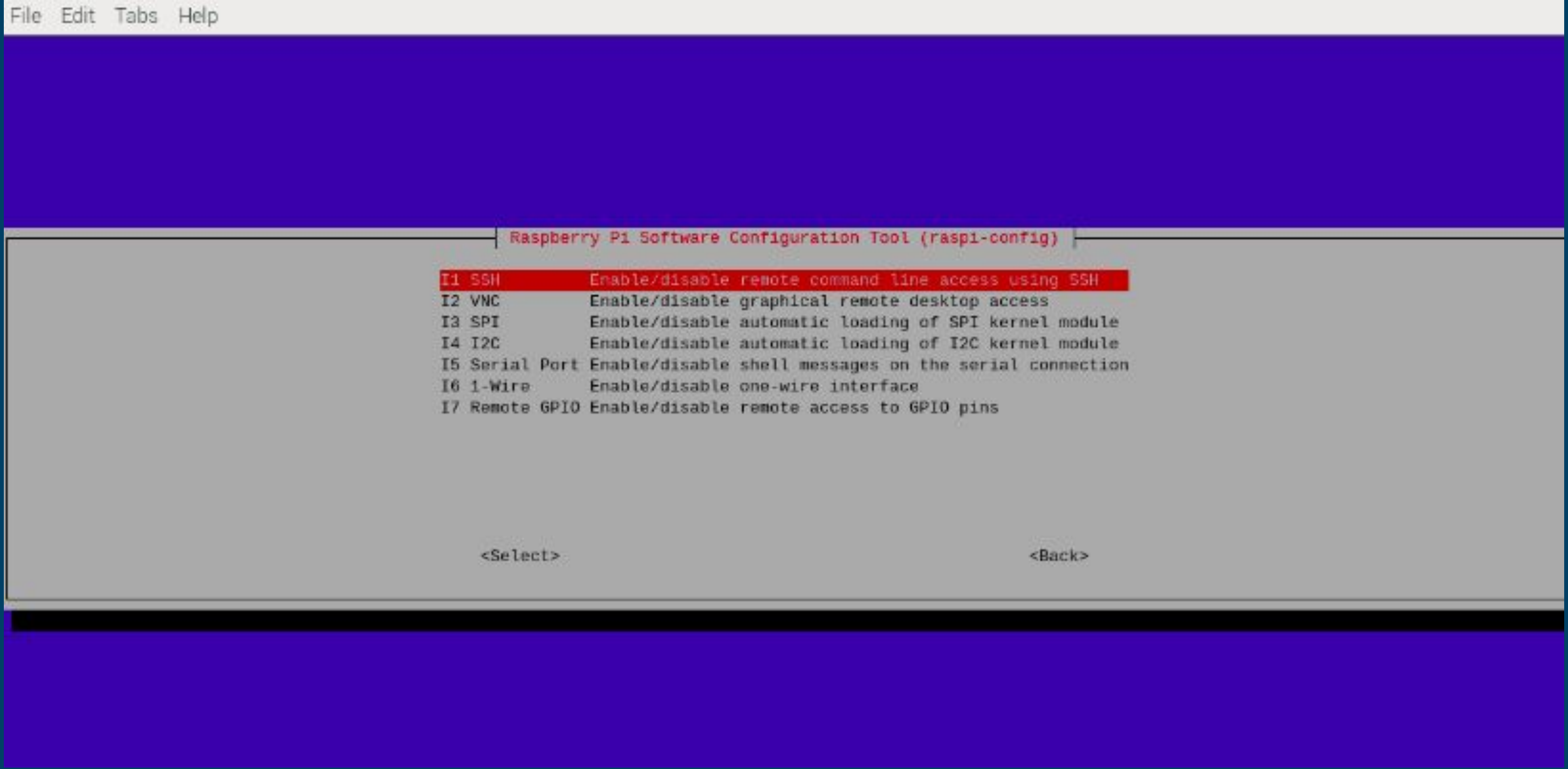
Raspberry Pi Settings



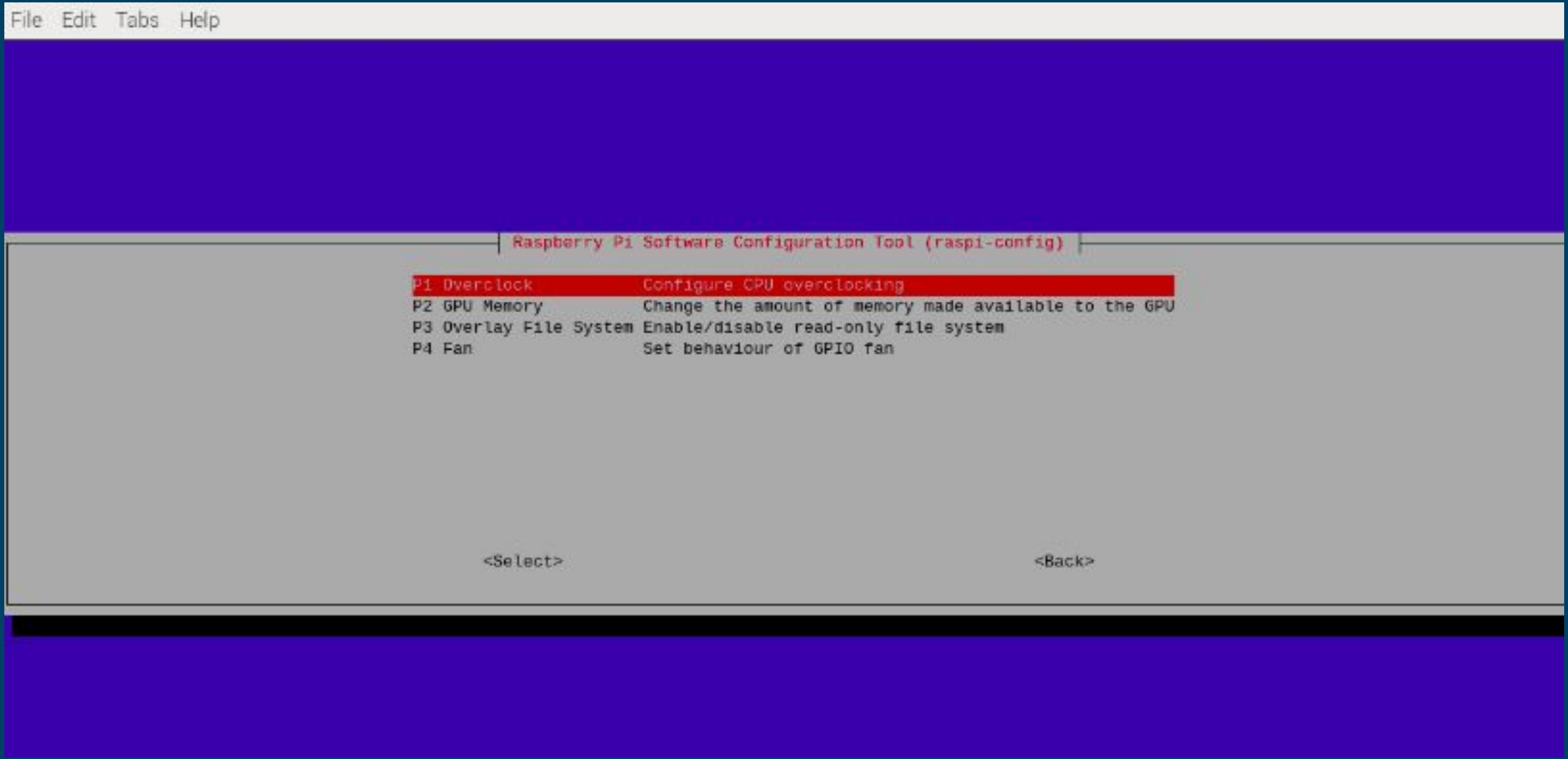
Raspberry Pi Settings



Raspberry Pi Settings



Raspberry Pi Settings



Raspberry Pi Settings

Raspberry Pi Settings We can modify these settings according to our requirements. There are many useful configurations that are beneficial for students.

More Modifications and customizations for Educational Purpose

- Modifying the bootloader to load your kernel.
- Customizing device drivers for hardware access. (We can do this if we get permission)
- Adapting GPU drivers or libraries for a simplified user interface.
- Tweaking system configuration to reduce resource usage.
- Implementing hardware-specific code for the Raspberry Pi's unique features.

Important Topic which can be Modified

- ## **Tweaking System Configuration**

  Tweaking the system configuration involves optimizing your OS for resource usage.

  We reduce resource usage by disabling unnecessary services, optimizing memory usage, and using minimal configurations.

  by using configuration files for the kernel and system services to make these adjustments

Important Topic which can be Modified

- **Adapting GPU drivers or libraries for a simplified user interface.**

  Do customizations and Modifications in existing GPU libraries

Important Topic which can be Modified

- **Implementing Hardware-Specific Code:**

-> Writing code specific to the Raspberry Pi's unique features or hardware components (e.g., VideoCore, camera module, or HATs).

-> Study official Raspberry Pi documentation and then Create hardware-specific code to enable and control these features in your OS.

Team Members

Gautam Kumar Mahar

Kanwar Raj

Akansha Madavi

Afzal Hussain

Thaksen Karote

Any Questions?