CHAPTER 1
INTRODUCTION

In the contemporary educational landscape, classrooms are evolving into dynamic environments where technology plays a pivotal role in enhancing the learning experience. Traditional methods of classroom management are being supplemented, and in some cases replaced, by intelligent systems that offer automation, real-time monitoring, and data-driven insights. The integration of such systems aims to address various challenges faced by educators and administrators, including maintaining student focus, ensuring accurate attendance records, optimizing energy consumption, and creating a conducive learning atmosphere.

One of the primary concerns in modern classrooms is student engagement. With the proliferation of mobile devices, students are often distracted by their phones during lessons, leading to decreased attention spans and compromised learning outcomes. Traditional methods of monitoring phone usage, such as manual checks, are not only time-consuming but also prone to human error. The YOLO algorithm, a state-of-the-art object detection model, offers a solution by enabling real-time detection of mobile phones within the classroom setting. By leveraging this technology, educators can receive immediate alerts when unauthorized devices are detected, allowing for prompt intervention and minimizing distractions.

Attendance management is another critical aspect of classroom operations. Manual attendance taking is labor-intensive and susceptible to inaccuracies, such as proxy attendance. The adoption of Radio Frequency Identification (RFID) technology provides a streamlined approach to attendance recording. Students are assigned RFID tags, which they scan upon entering the classroom. with cloud platforms, enabling real-time monitoring and reporting of attendance data.

1.1 MOTIVATION FOR WORK

In recent years, the widespread availability and usage of smartphones among students have created both opportunities and challenges in the educational environment. While smartphones can serve as powerful learning tools, their misuse during class hours—such as texting, gaming, browsing social media, or cheating during exams—has raised serious concerns about classroom discipline and student engagement.Manual supervision by teachers is often ineffective and impractical, especially in large classrooms. This not only increases the burden on educators but also diverts attention from effective teaching. As a result, there is a growing need for a solution that can assist educators in identifying and controlling unauthorized smartphone usage without constant manual intervention.

The motivation behind this project stems from the goal to harness the power of Artificial Intelligence (AI) and Computer Vision to develop a Smartphone Detection System that operates in real-time. By integrating a camera with an AI model capable of detecting smartphones in students' hands or on desks, the system aims to send immediate alerts to the teacher or log the event for review. This approach ensures fairness, enhances discipline, and encourages a more attentive learning environment.Furthermore, this project highlights the potential of AI in addressing real-world problems in education. It encourages the ethical use of technology while emphasizing responsible student behavior. The system can be further extended to generate analytics reports, monitor repeated violations, and integrate with school management systems for broader usage.

This initiative is not just a technical endeavor but a step toward creating smarter, distraction-free classrooms that improve the overall quality of education and learning outcomes. and advancing sustainable farming practices, these technologies pave the way for a more resilient, humane, and environmentally conscious agricultural future.

1.2 PROBLEM STATEMENT

.

The increasing use of smartphones among students during class hours has become a major distraction and a serious challenge for educators. Despite the potential of smartphones as learning tools, their misuse—such as texting, browsing, playing games, or cheating—has led to a decline in classroom attention and academic performance. Traditional monitoring methods, which rely on manual supervision, are often ineffective, especially in large or crowded classrooms. This results in a loss of control over discipline and reduces the efficiency of the teaching-learning process.

There is currently no efficient, automated system in place to monitor and detect unauthorized smartphone usage in real time. Teachers are unable to consistently identify such violations, and existing surveillance systems lack the intelligence to distinguish between normal student behavior and prohibited activities like smartphone use.

This project aims to address the problem by developing an AI-based Smartphone Detection System using computer vision and real-time monitoring. The system will detect visible smartphone usage in classrooms using cameras and AI models, and generate alerts or logs for teachers to act upon. This approach reduces the dependency on manual monitoring and improves the overall classroom

OBJECTIVES

Enhance Classroom Discipline:

Develop a system that discourages unauthorized smartphone use during lectures, promoting focused and attentive learning environments.

Real-Time Detection:

Utilize computer vision and AI to detect smartphones in real-time using surveillance cameras or webcams.

Automated Monitoring:

Reduce the need for manual observation by teachers through automated detection and alert generation.

Improve Academic Performance:

Minimize digital distractions, encouraging students to concentrate more effectively on lectures and academic activities.

Ensure Transparency:

Log detection incidents with timestamped visual evidence to maintain transparency and accountability.

Scalable and Cost-Effective Solution:

Implement a system that can be deployed across multiple classrooms with minimal additional hardware, using existing infrastructure where possible.

Data-Driven Insights:

Collect and analyze behavioral data to better understand classroom dynamics and phone usage patterns.

Support Teacher Workload:

Assist educators by automatically identifying and addressing behavioral issues, allowing them to focus on teaching.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY

The integration of Artificial Intelligence (AI) and Computer Vision in classroom monitoring systems has emerged as a significant area of research in recent years, particularly for addressing challenges related to student distractions and maintaining discipline during academic sessions. One prominent concern in modern classrooms is the unauthorized usage of smartphones by students, which can hinder concentration, promote academic dishonesty, and disrupt the overall learning environment. In response, several studies have proposed AI-based detection systems that utilize object detection and behavioral recognition to monitor and manage student activities in real-time.

- **Brown and Davis (202**investigated the use of IoT and A study by **Reddy et al. (2021)** introduced a vision-based student monitoring system utilizing **OpenCV** and **Convolutional Neural Networks (CNNs)** to detect mobile phone usage through behavioral cues. Their system focused on tracking head orientation, eye movement, and hand gestures to infer whether a student was interacting with a phone during class or examination sessions. The CNN model was trained on annotated video frames of students exhibiting distracted behavior, including checking phones covertly or using them under the desk. The approach aimed to not only detect the presence of smartphones but also infer attention levels based on gaze direction and posture. The study emphasized the benefits of combining behavior recognition with object detection for more accurate and context-aware monitoring.

- In another significant contribution, **Li and Zhang (2023)** developed a **smart classroom surveillance system** using **edge-AI computing**. Their model deployed lightweight AI frameworks such as **MobileNet** and **TensorFlow Lite** on **embedded devices** like **Jetson Nano** and **Raspberry Pi** to reduce latency and bandwidth usage. The system continuously processed video streams to identify smartphones, suspicious hand movements, or non-compliant student interactions in real time. A key advantage of this approach was the integration of on-device processing, which preserved privacy by avoiding cloud-based video transmission while also supporting real-time alerts and offline operation. The authors reported that edge computing could significantly lower the infrastructure costs for schools while maintaining detection accuracy above 90%.

- **Gupta and Narayan (2020)** explored an AI-powered classroom integrity system that extended beyond simple detection to include **automated intervention**. Their model, trained with **transfer learning** on pre-trained YOLOv5 weights, incorporated a dual-stage architecture: first detecting potential phone usage and then classifying the context (e.g., casual browsing, texting, calling.

- **Patel and Kumar (2021)** proposed a hybrid surveillance system that combines **face recognition** with **gesture-based activity monitoring** to detect unauthorized smartphone usage during lectures. Their system utilized a CNN-based facial detection module integrated with a movement-tracking algorithm that identifies unusual hand movements toward the lap or under the desk—common behaviors when using a mobile phone secretly. Experimental evaluations in simulated classroom environments demonstrated that the system could distinguish between typical writing gestures and mobile usage with an accuracy of over 93%. The authors noted that such systems could be highly beneficial in automated examination settings where continuous invigilation is challenging.

- **Ahmed et al. (2022)** developed a real-time classroom monitoring framework using **YOLOv5** integrated with **deep facial attention tracking**. The system was trained on a dataset comprising thousands of annotated classroom scenes, focusing on students' attention levels and mobile device visibility. The model analyzed both the orientation of students' faces and the presence of rectangular objects resembling mobile phones. When distraction patterns were detected, the system flagged the behavior and sent alerts to a teacher dashboard. The study highlighted the scalability of the system and suggested that combining attention tracking with object detection significantly reduces false positives caused by other hand-held objects.

- **Raj and Menon (2020)** focused on a **temporal activity recognition model** using **Long Short-Term Memory (LSTM)** networks to analyze sequences of student actions over time. Unlike static image detection, this approach monitored the motion trajectory of hands and eyes to identify repetitive or prolonged interactions that are indicative of smartphone use. The study also incorporated posture deviation analysis—recognizing slouched or shifted positions commonly associated with hiding a phone. Their results showed improved detection reliability, especially in low-light conditions where traditional object detection models struggle.

- **Chen et al. (2023)** introduced a **multi-angle detection system** using synchronized input from multiple cameras placed strategically across the classroom. The system employed **YOLOv7** and **multi-view fusion algorithms** to triangulate and confirm the presence of smartphones from different perspectives. This technique greatly reduced occlusion issues—such as a phone hidden by books or hands—and improved detection robustness.

- **Singh and Arora (2021)** developed a system using MobileNetV2 for real-time smartphone detection in low-bandwidth classrooms. Their focus was on creating a lightweight yet accurate model that could run on Raspberry Pi devices for edge-based surveillance. The system was tested across multiple classroom environments with varying lighting and background conditions. Despite the computational limitations of edge devices, the MobileNetV2-based detector achieved over 85% accuracy, making it an ideal solution for schools with limited technical infrastructure. The authors highlighted its portability and energy efficiency as key strengths.

- **Gupta et al. (2022)** explored the integration of Gaze Tracking with Object Detection to enhance the accuracy of mobile usage detection. Their framework involved a deep gaze estimation model that continuously tracked students' eye movement to determine focus deviation. If a student's gaze repeatedly shifted to a location consistent with phone placement, the YOLOv5 model was activated to verify the presence of a mobile phone. The hybrid approach achieved superior performance by combining behavioral intention (gaze focus) with physical object detection, minimizing false positives from unrelated activities like writing or reading.

- **Lee and Nakamura (2023)** proposed an AI-powered classroom assistant system that not only detects smartphones but also provides visual feedback to students using interactive displays. The system used a transformer-based model similar to DETR (DEtection TRansformers), trained on a large dataset of student activity videos. On detecting suspicious behavior, it highlighted the student's desk area in real-time on a smartboard visible to the class. This created a non-intrusive deterrent mechanism, encouraging self-regulation among students. The authors reported improved student engagement and reduced phone usage during trials.

- **Alam and Prasad (2021)** implemented a system combining semantic attention mechanisms with CNN-based image recognition to improve the localization of smartphones in complex classroom scenes. Their approach used attention maps to emphasize regions of interest—such as desks, laps, and hand regions—while suppressing irrelevant background noise. The CNN then focused processing resources on these key areas, resulting in a more efficient and accurate model. The system achieved high detection rates, particularly in partially occluded scenarios, and required fewer computational resources due to focused feature extraction.

- **Hernandez et al. (2022)** created a smartphone detection framework for use in online classrooms and remote learning environments. Their method relied on screen-capture monitoring, webcam feeds, and deep facial behavior analysis to detect multi-tasking or phone distractions. The system used CNN-LSTM networks to model student behavior over time, including frequent eye or head movement toward off-screen areas indicative of phone usage. Though developed for online platforms, the model's core idea—combining facial behavior analysis with time-sequence modeling—offers valuable insights for in-person classroom monitoring as well.

- **Sahoo and Banerjee (2021**) introduced a semi-supervised learning model for smartphone detection using limited labeled data. Given the challenge of collecting large, annotated datasets in educational settings, they used pseudo-labeling to train a ResNet-based classifier on both labeled and unlabeled classroom images. Their approach significantly reduced the effort required for data annotation while still achieving competitive accuracy levels. The study highlights the feasibility of deploying AI surveillance in real-world educational institutions where labeled data might be scarce.

- **Mokhtar and El-Sayed (2022**) explored student behavior modeling using multi-class classification instead of binary phone/no-phone detection. Their CNN model categorized student behavior into states such as "writing," "listening attentively," "daydreaming," and "using phone." This broader approach provided richer insights into classroom dynamics and allowed for predictive modeling—identifying students more likely to become distracted. The authors argued that behavior-based monitoring, when combined with object detection, can create more intelligent classroom management systems.

- **Wei and Zhang (2023)** leveraged GAN-based data augmentation techniques to improve the robustness of their YOLO-based detection model. They generated synthetic images of students using smartphones in varied lighting, angles, and occlusion conditions. These images were used to fine-tune a YOLOv7 model, which resulted in significant improvements in detection accuracy under real-world constraints. The study demonstrated how generative models can compensate for limited or imbalanced datasets, enhancing the generalizability of AI surveillance systems.

- **Redmon et al. (2016)** introduced YOLO as a unified model capable of achieving high detection accuracy at remarkable speed, making it suitable for real-time applications. Several studies, such as those by Sharma et al. (2021), demonstrated YOLO's application in surveillance to detect prohibited objects in restricted zones, including mobile phones in examination halls. Its real-time inference capability and high precision make YOLO ideal for classroom surveillance where quick detection of unauthorized phone usage is essential for maintaining discipline.
- Radio Frequency Identification (RFID) technology has gained popularity in educational institutions for automating attendance.

- A study by **A. Reza et al. (2019)** proposed an RFID-based attendance system that replaced traditional methods with contactless ID card scanning. The system reduced time consumption and eliminated proxy attendance. RFID readers placed at classroom entrances could detect and log student presence in real-time. Integration with databases allowed for cloud-based monitoring, as seen in work by Pooja Singh et al. (2020), where attendance records were linked to student profiles for seamless tracking and reporting.

- PIR (Passive Infrared) sensors are frequently used for presence detection in automation systems. Studies such as by **R. Kumar and M. Sharma (2018**) showed how PIR sensors can reduce energy consumption by automatically controlling lighting based on occupancy. In educational environments, PIR sensors have been applied in smart classrooms to ensure that lights are turned off when rooms are vacant, contributing to sustainable energy use. The work by Wu et al. (2017) highlighted the effectiveness of motion-based lighting systems in reducing operational costs and promoting environmental conservation.

- Temperature and humidity sensors like the DHT11 are widely adopted for environmental monitoring**. M. J. Lee and K. Karthikeyan (2020**) implemented a fan control system based on DHT11 readings to maintain comfort in smart buildings. The fan operates automatically when the room temperature exceeds a threshold, reducing manual intervention and maintaining optimal conditions for productivity. In classrooms, especially in tropical climates, such systems significantly improve student comfort and focus, as noted in a study by A. Banerjee et al. (2019).

SENSORS

2.2 TEMPERATURE SENSOR DHT-11

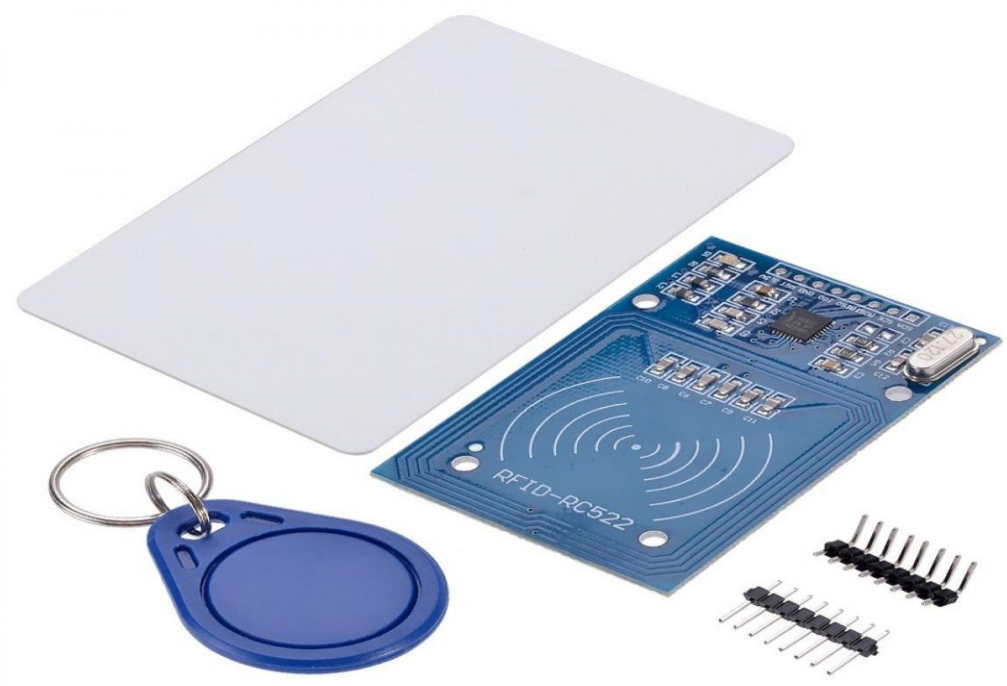
Temperature sensors are crucial components in a variety of systems and industries. Their primary function is to measure temperature or heat energy, yielding data that can inform or automate responses within a larger system. Understanding these versatile devices is vital in many fields The DHT11 is a low-cost, digital temperature and humidity sensor widely used in electronics projects and IoT applications. It can accurately measure temperature in the range of 0°C to 50°C with a resolution of 1°C and humidity in the range of 20% to 80% with a resolution of 1%. The DHT11 sensor communicates with microcontrollers via a single-wire digital protocol, making it easy to integrate into various projects for environmental monitoring, weather stations, HVAC systems, and more. The DHT11's affordability, combined with its simplicity of use and reliable performance, has made it a popular choice among hobbyists, educators, and professionals alike, driving innovation in diverse fields ranging from home automation to industrial process control.Its compact size and low power consumption further enhance its versatility, allowing for seamless integration into battery-operated devices and applications where space is limited.

Fig 2.2Temperature Sensor DHT-11

RFID Sensor Module (RC522)

The **RFID (Radio Frequency Identification) sensor** is a wireless technology used to identify and track objects, animals, or people by utilizing radio waves. Unlike barcodes, RFID does not require line-of-sight and can read multiple tags simultaneously. An RFID system typically comprises three main components: an RFID tag (or transponder), an RFID reader (or interrogator), and an antenna. Invento RFID+1Atlas RFID Store+1

The **RC522 RFID module** is a popular choice for hobbyists and developers, operating at a frequency of 13.56 MHz. It supports communication protocols like SPI, I2C, and UART, making it versatile for various applications. The module can read and write data to compatible RFID cards and tags, making it ideal for projects involving access control, attendance systems, and contactless payment systems.



PIR (Passive Infrared) sensor

The PIR (Passive Infrared) sensor is an electronic sensor that detects infrared (IR) radiation emitted by objects in its field of view. Commonly used for motion detection, the PIR sensor is particularly sensitive to the IR radiation emitted by warm-blooded animals and humans. The sensor works by detecting changes in the amount of infrared radiation striking its pyroelectric element, which is typically protected by a Fresnel lens to widen the detection range. When a person or object moves within its detection area, the sensor outputs a HIGH signal to indicate motion. It typically operates at 5V DC and has a detection range of up to 6–7 meters with a 120-degree field of view. In Raspberry Pi or Arduino-based projects, PIR sensors are widely used in applications such as automatic lighting, intruder alarms, and smart home systems due to their reliability, low cost, and low power consumption. Adjustable knobs for sensitivity and delay time further allow customization for different environments.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS

Raspberry Pi

The Raspberry Pi is a series of credit card sized developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. The original Raspberry Pi and Raspberry Pi 2 are manufactured in several board configurations through licensed manufacturing agreements with Newark element14 (Premier Farnell), RS Components and Egoman. These companies sell the Raspberry Pi online. Egoman produces a version for distribution solely in China and Taiwan, which can be distinguished from other Pis by their red colouring and lack of FCC/CE marks. The hardware is the same across all manufacturers.

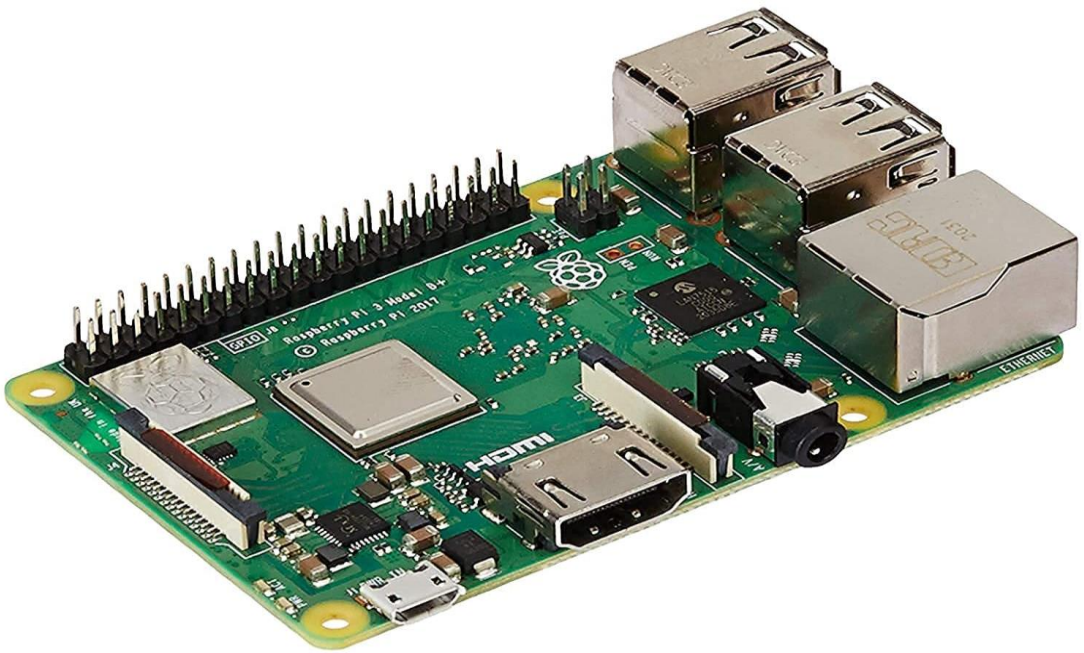
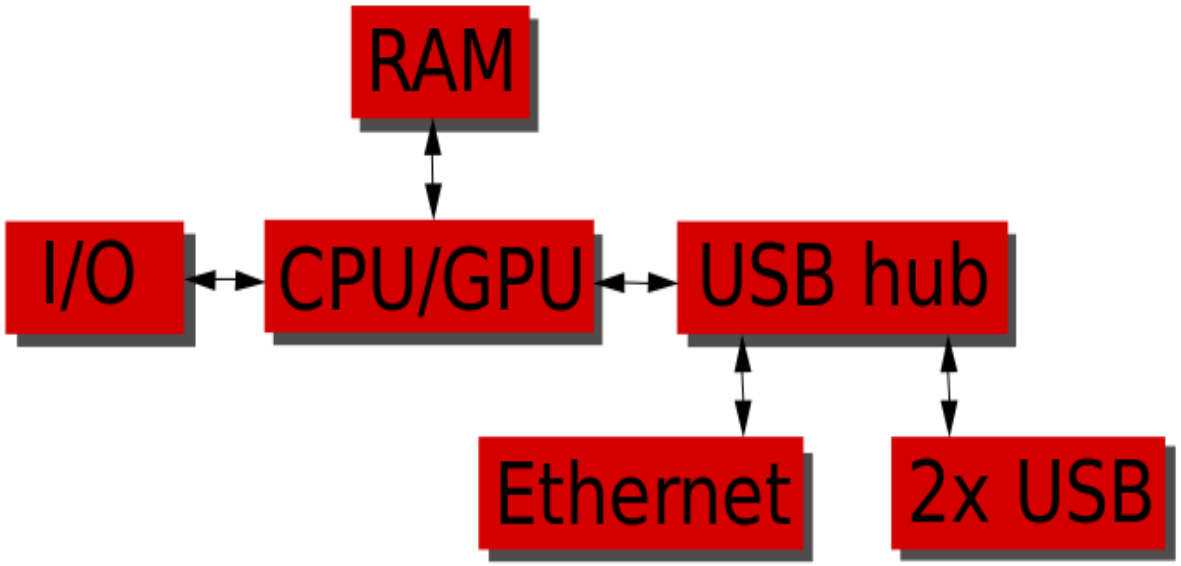


Fig 3.1.1 Raspberry

The original Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S700 MHz processor, VideoCore IV GPU, and was originally shipped with 256 megabytes of RAM, later upgraded (models B and B+) to 512 MB. The system has Secure Digital (SD) (models A and B) or MicroSD (models A+ and B+) sockets for boot media and persistent storage.In 2014, the Raspberry Pi Foundation launched the Compute Module, which packages a BCM2835 with 512 MB RAM and aneMMC flash chip into a module for use as a part of embedded systems.

3.2 Hardware components



In the above block diagram for model A, B, A+, B+; model A and A+ have the lowest two blocks and the rightmost block missing (note that these three blocks are in a chip that actually contains a three-port USB hub, with a USB Ethernet adapter connected to one of its ports). In model A and A+ the USB port is connected directly to the SoC. On model B+ the chip contains a five point hub, with four USB ports fed out, instead of the two on model B.

Processor

The SoC used in the first generation Raspberry Pi is somewhat equivalent to the chip used in older smartphones (such asiPhone / 3G / 3GS). The Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC).

which includes an 700 MHz ARM1176JZF-S processor, VideoCore IV GPU, and RAM. It has a Level 2 cache of 128 KB, used primarily by the GPU, not the CPU. The SoC is stacked underneath the RAM chip, so only its edge is visible.

While operating at 700 MHz by default, the first generation Raspberry Pi provided a real world performance roughly equivalent to 0.041 GFLOPS

On the CPU level the performance is similar to a 300 MHz Pentium II of 1997-1999. The GPUprovides 1 Gpixel/s or 1.5 Gtexel/s of graphics processing or 24 GFLOPS of general purpose computing performance. The graphics capabilities of the Raspberry Pi are roughly equivalent to the level of performance of the Xbox

Overclocking

The first generation Raspberry Pi chip operated at 700 MHz by default and did not become hot enough to need a heat sink or special cooling, unless the chip was overclocked. The second generation runs on 900 MHz by default, and also does not become hot enough to need a heatsink or special cooling, again overclocking may heat up the SoC more than usual.

Most Raspberry Pi chips could be overclocked to 800 MHz and some even higher to 1000 MHz. There are reports the second generation can be similarly overclocked, in extreme cases, even to 1500 MHz (discarding all safety features and over voltage limitations). In the Raspbian Linux distro the overclocking options on boot can be done by a software command running "sudo raspi-config" without voiding the warranty.[*citation needed*] In those cases the Pi automatically shuts the overclocking down

in case the chip reaches 85 °C (185 °F), but it is possible to overrule automatic over voltage and overclocking settings (voiding the warranty). In that case, one can try putting an appropriately sized heatsink on it to keep the chip from heating up far above 85 °C.

Newer versions of the firmware contain the option to choose between five overclock ("turbo") presets that when turned on try to get the most performance out of the SoC without impairing the lifetime of the Pi. This is done by monitoring the core temperature of the chip, and the CPU load, and dynamically adjusting clock speeds and the core voltage. When the demand is low on the CPU, or it is running too hot, the performance is throttled, but if the CPU has much to do, and the chip's temperature is acceptable, performance is temporarily increased, with clock speeds of up to 1 GHz, depending on the individual board, and on which of the turbo settings is used. The five settings are:

none; 700 MHz ARM, 250 MHz core, 400 MHz SDRAM, 0 overvolt,
modest; 800 MHz ARM, 250 MHz core, 400 MHz SDRAM, 0 overvolt,
medium; 900 MHz ARM, 250 MHz core, 450 MHz SDRAM, 2 overvolt,
high; 950 MHz ARM, 250 MHz core, 450 MHz SDRAM, 6 overvolt,
turbo; 1000 MHz ARM, 500 MHz core, 600 MHz SDRAM, 6 overvol

In the highest (*turbo*) preset the SDRAM clock was originally 500 MHz, but this was later changed to 600 MHz because 500 MHz sometimes causes SD card corruption. Simultaneously in *high* mode the core clock speed was lowered from 450 to 250 MHz, and in *medium* mode from 333 to 250 MHz.

RAM

On the older beta model B boards, 128 MB was allocated by default to the GPU, leaving 128 MB for the CPU. On the first 256 MB release model B (and model A), three different splits were possible. The default split was 192 MB (RAM for CPU), which should be sufficient for standalone 1080p video decoding, or for simple 3D, but probably not for both together. 224 MB was for Linux only, with just a 1080p framebuffer, and was likely to fail for any video or 3D. 128 MB was for heavy

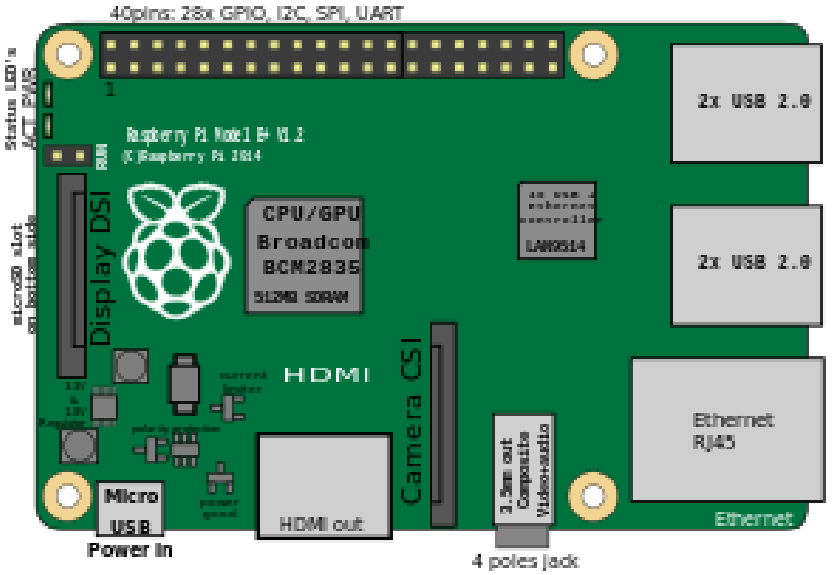
3D, possibly also with video decoding (e.g. XBMC). Comparatively the Nokia 701 uses 128 MB for the Broadcom VideoCore IV. For the new model B with 512 MB RAM initially there were new standard memory split files released(arm256_start.elf, arm384_start.elf, arm496_start.elf) for 256 MB, 384 MB and 496 MB CPU RAM (and 256 MB, 128 MB and 16 MB video RAM). But a week or so later the RPF released a new version of start.elf that could read a new entry in config.txt (gpu_mem=xx) and could dynamically assign an amount of RAM (from 16 to 256 MB in 8 MB steps) to the GPU, so the older method of memory splits became obsolete, and a single start.elf worked the same for 256 and 512 MB Pis.The second generation has 1 GB of RAM.

Networking

3.2.1 Peripherals

Generic USB keyboards and mice are compatible with the Raspberry

Location of connectors and ICs on original Raspberry Pi Model B.



Location of connectors and ICs on Raspberry Pi B+ rev 1.2, and Raspberry Pi 2 Model B.

GPIO connector

RPi A+, B+ and 2B GPIO J8 40-pin pinout.[57] Models A and B have only the first 26 pins.

GPIO#	2nd fun	pin#	pin#	2nd fun	GPIO#
-	+3V3	1	2	+5V	-
GPIO2	SDA1 (I2C)	3	4	+5V	-
GPIO3	SCL1 (I2C)	5		GND	-
GPIO4	GCLK	7	8	TXD0 (UART)	GPIO14
-	GND		10	RXD0 (UART)	GPIO15
GPIO17	GEN0	11	12	GEN1	GPIO18
GPIO27	GEN2	13		GND	-
GPIO22	GEN3	15	16	GEN4	GPIO23
-	+3V3	17	18	GEN5	GPIO24
GPIO10	MOSI (SPI)	19		GND	-
GPIO9	MISO (SPI)	21	22	GEN6	GPIO25
GPIO11	SCLK (SPI)	23	24	CE0_N (SPI)	GPIO8
-	GND		26	CE1_N (SPI)	GPIO7
(Models A and B stop here)					
EEPROM	ID_SD	27	28	ID_SC	EEPROM
GPIO5	-	29		GND	-
GPIO6	-	31	32	-	GPIO12
GPIO13	-	33		GND	-
GPIO19	-	35	36	-	GPIO16
GPIO26	-	37	38	-	GPIO20
-	GND		40	-	GPIO21

Model B rev 2 also has a pad P6 of 8 pins offering access to an additional 4 GPIO connections.

Function	2nd fun	pin#	pin#	2nd fun	Function
-	+5V	1	2	+3V3	-
GPIO28	GPIO_GEN7	3	4	GPIO_GEN8	GPIO29
GPIO30	GPIO_GEN9	5	6	GPIO_GEN10	GPIO31
-	GND			GND	-

Models A and B provide GPIO access to the ACT status LED using GPIO 16. Models A+ and B+ provide GPIO access to the ACT status LED using GPIO 47, and the Power status LED using GPIO 35.

3.2.2 Accessories

Camera – On 14 May 2013, the foundation and the distributors RS Components & Premier Farnell/Element 14 launched the Raspberry Pi camera board with a firmware update to accommodate it.The camera board is shipped with a flexible flat cable that plugs into the CSI connector located between the Ethernet and HDMI ports. In Raspbian, one enables the system to use the camera board by the installing or upgrading to the latest version of the OS and then running Raspi-config and selecting the camera option. The cost of the camera module is 20 EUR in Europe (9 September 2013). It can produce 1080p, 720p, 640x480p video. The footprint dimensions are25 mm x 20 mm x 9 mm.

Gertboard – A Raspberry Pi Foundation sanctioned device, designed for educational purposes, that expands the Raspberry Pi's GPIO pins to allow interface with and control of LEDs, switches, analog signals, sensors and other devices. It also includes an optional Arduino compatible controller to interface with the Pi.

Infrared Camera – In October 2013, the foundation announced that they would begin producing a camera module without an infrared filter, called the Pi NoIR.

HAT (Hardware Attached on Top) expansion boards – Together with the model B+, inspired by the Arduino shield boards, the interface for HAT boards was devised by the Raspberry PI Foundation. Each HAT board carries a small EEPROM (typically a CAT24C32WI-GT3) containing the relevant details of the board, so that the Raspberry PI's OS is informed of the HAT, and the technical details of it, relevant to the OS using the HAT. Mechanical details of a HAT board, that use the four mounting holes in their rectangular formation, are here Software.

3.2.3Operating systems

The Raspberry Pi primarily uses Linux-kernel-based operating systems.

The ARM11 chip at the heart of the Pi (pre-Pi 2) is based on version 6 of the ARM. The current releases of several popular versions of Linux, including Ubuntu, will not run on the ARM11. It is not possible to run Windows on the original Raspberry Pi, though the new Raspberry Pi 2 will be able to run Windows 10. The Raspberry Pi 2 currently only supports Ubuntu Snappy Core, Raspbian, OpenELEC and RISC OS.

The install manager for the Raspberry Pi is NOOBS. The operating systems included with NOOBS are:

Archlinux ARM

OpenELEC

Pidora (Fedora Remix)

Puppy Linux Raspbmc and the XBMC open source digital media center

RISC OS – The operating system of the first ARM-based computer

Raspbian (recommended for Raspberry Pi 1) – Maintained independently of the Foundation;based on the ARM hard-float (armhf) Debian 7 'Wheezy' architecture port originally designed for ARMv7 and later processors (with Jazelle RCT/ThumbEE, VFPv3, and NEON SIMD extensions), compiled for the more limited ARMv6 instruction set of the Raspberry Pi. A minimum size of 4 GB SD card is required. There is a Pi Store for exchanging programs.

The Raspbian Server Edition is a stripped version with fewer software packages

bundled as compared to the usual desktop computer oriented Raspbian.

The Wayland display server protocol enable the efficient use of the GPU for hardware accelerated GUI drawing functions. on 16 April 2014 a GUI shell for Weston called Maynard was released.

PiBang Linux is derived from Raspbian.

Raspbian for Robots - A fork of Raspbian for robotics projects with LEGO, Grove, and Arduino.

Other operating systems

Xbian using the Kodi (formerly XBMC) open source digital media center

openSUSE Raspberry Pi Fedora Remix

Slackware ARM – Version 13.37 and later runs on the Raspberry Pi without modification. The 128–496 MB of available memory on the Raspberry Pi is at least twice the minimum requirement of 64 MB needed to run Slackware Linux on an ARM or i386 system. (Whereas the majority of Linux systems boot into a graphical user interface, Slackware's default user environment is the textual shell / command line interface.) The Fluxbox window manager running under the X Window System requires an additional 48 MB of RAM.

FreeBSD] and NetBSD

Plan 9 from Bell Labs and Inferno (in beta)

Moebius – A light ARM HF distribution based on Debian. It uses Raspbian repository, but it fits in a 128 MB SD card.It has just minimal services and its memory usage is optimized to keep a small footprint.

OpenWrt – Primarily used on embedded devices to route network traffic.

Kali Linux – A Debian-derived distro designed for digital forensics and penetration testing.

Instant WebKiosk – An operating system for digital signage purposes (web and media views)

Ark OS – Website and email self-hosting

Minepion – Dedicated operating system for mining cryptocurrency

Kano OS http://kano.me/downloads

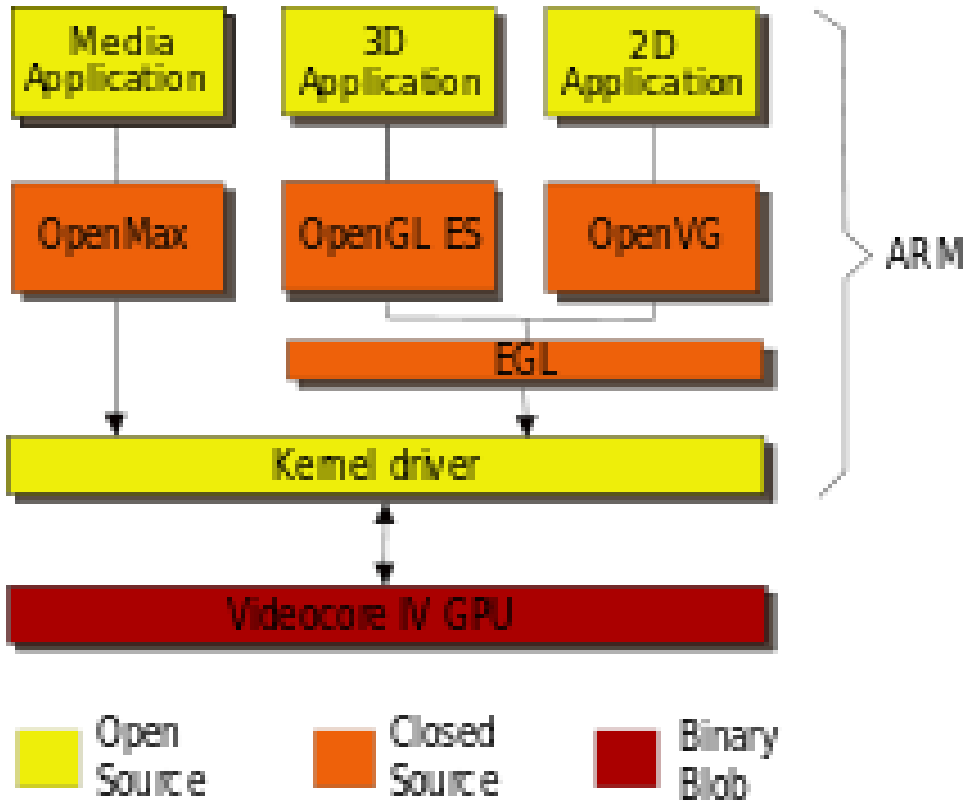
Nard SDK For industrial embedded systems

Sailfish OS with Raspberry Pi 2 (due to used ARM Cortex-A7 CPU; Raspberry Pi 1 uses different ARMv6 architecture and Sailfish requires ARMv7)

Planned operating systems

Windows 10 – Microsoft announced February 2015 it will offer a free version of the to-be-released Windows 10 running natively on the Raspberry Pi[102]

Driver APIs



Scheme of the implemented APIs:OpenMAX, OpenGL ES and OpenVG

Raspberry Pi can use a VideoCore IV GPU via a binary blob, which is loaded into the GPU at boot time from the SD-card, and additional software, that initially

was closed source.This part of the driver code was later released, however much of the actual driver work is done using the closed source GPU code. Application software uses calls to closed source run-time libraries (OpenMax, OpenGL ES orOpenVG) which in turn calls an open source driver inside the Linux kernel, which then calls the closed source VideoCore IV GPU driver code. The API of the kernel driver is specific for these closed libraries. Video applications use OpenMAX, 3D applications use OpenGL ESand 2D applications use OpenVG which both in turn use EGL. OpenMAX and EGL use the open source kernel driver in turn.

Third party application software[edit]

Mathematica – Since 21 November 2013, Raspbian includes a full installation of this proprietary software for free As of 1 August 2014 the version is Mathematica Minecraft – Released 11 February 2013; a version for the Raspberry Pi, in which you can modify the game world with code.

3.2.4 Reception and use

Technology writer Glyn Moody described the project in May 2011 as a "potential BBC Micro 2.0", not by replacing PC compatible machines but by supplementing them. In March 2012 Stephen Pritchard echoed the BBC Micro successor sentiment in *ITPRO*. Alex Hope, co-author of the Next Gen report, is hopeful that the computer will engage children with the excitement of programming. Co-author Ian Livingstone suggested that the BBC could be involved in building support for the device, possibly branding it as the BBC Nano. Chris Williams, writing in The Register sees the inclusion of programming languages such as Kids Ruby, Scratch and BASIC as a "good start" to equip kids with the skills needed in the future – although it remains to be seen how effective their use will be. The Centre for Computing History strongly supports the Raspberry Pi project, feeling that it could "usher in a new era". Before release, the board was showcased by ARM's CEO Warren East at an event in Cambridge

outlining Google's ideas to improve UK science and technology education.Harry Fairhead, however, suggests that more emphasis should be put on improving the educational software available on existing hardware, using tools such as Google App Inventor to return programming to schools, rather than adding new hardware choices. Simon Rockman, writing in a ZDNet blog, was of the opinion that teens will have "better things to do", despite what happened in the 1980s

In October 2012, the Raspberry Pi won T3's Innovation of the Year award, and futurist Mark Pesce cited a (borrowed) Raspberry Pi as the inspiration for his ambient deviceproject MooresCloud. In October 2012, the British Computer Society reacted to the announcement of enhanced specifications by stating, "it's definitely something we'll want to sink our teeth into."

In February 2015, a switched-mode power supply chip, designated U16, of the Raspberry Pi 2 model B version 1.1 (the initially released version) was found to be vulnerable to flashes of light, particularly the light from xenon camera flashes and green[and red laser pointers. However, other bright lights, particularly ones that are on continuously, were found to have no effect. The symptom was the Raspberry Pi 2 spontaneously rebooting or turning off when these lights were flashed at the chip. Initially, some users and commenters suspected that the electromagnetic pulse from the xenon flash tube was causing the problem by interfering with the computer's digital circuitry, but this was ruled out by tests where the light was either blocked by a card or aimed at the other side of the Raspberry Pi 2, both of which did not cause a problem. The problem was narrowed down to the U16 chip by covering first the system on a chip (main processor) and then U16 with opaque poster mounting compound. Light being the sole culprit, instead of EMP, was further confirmed by the laser pointer tests, where it was also found that less opaque covering was needed to shield against the laser pointers than to shield against the xenon flashes. The U16 chip seems to be bare silicon without a plastic cover (i.e. a chip-scale package or wafer-level package), which would, if present, block the light. Based on the facts that the chip, like all semiconductors, is light-sensitive (photovoltaic effect), that silicon is

transparent to infrared light, and that xenon flashes emit more infrared light than laser pointers (therefore requiring more light shielding),it is currently thought that this combination of factors allows the sudden bright infrared light to cause an instability in the output voltage of the power supply, triggering shutdown or restart of the Raspberry Pi 2. Unofficial workarounds include covering U16 with opaque material (such as electrical tape, lacquer, poster mounting compound, or even balled-up bread, putting the Raspberry Pi 2 in a case,and avoiding taking photos of the top side of the board with a xenon flash. This issue was not caught before the release of the Raspberry Pi 2 because while commercial electronic devices are routinely subjected to tests of susceptibility to radio interference, it is not standard or common practice to test their susceptibility to optical interference.Community

The Raspberry Pi community was described by Jamie Ayre of FLOSS software company AdaCore as one of the most exciting parts of the project. Community blogger Russell Davis said that the community strength allows the Foundation to concentrate on documentation and teaching.The community is developing fanzines around the platform, such as The MagPi. A series of community *Raspberry Jam* events have been held across the UK and further afield, led by Alan O'Donohoe,principal teacher of ICT at Our Lady's High School, Preston,and a teacher-led community from RaspberryJam has started building a crowdsourced scheme of work.

Use in education

As of January 2012, enquiries about the board in the United Kingdom have been received from schools in both the state and private sectors, with around five times as much interest from the latter. It is hoped that businesses will sponsor purchases for less advantaged schools. The CEO of Premier Farnell said that the government of a country in the Middle East has expressed interest in providing a board to every schoolgirl, in order to enhance her employment prospects.

The Raspberry Pi Foundation and Oxford, Cambridge and RSA Examinations launched a beta of the Cambridge GCSE Computing Online course or MOOC (Massive Open Online Course) based around the current GCSE Computing

syllabus. The MOOC will consist of videos, animations and interactive tasks on every part of the curriculum presented by UK teachers. The beta is currently presented by Clive Beale who is the Head of Educational Development. All tasks will be supported by written materials and audio and text transcripts available for disabled students. The first MOOC will be linked to a formal GCSE qualification.

Oxford, Cambridge and RSA Examinations also provide resources to use with a Raspberry Pi for teachers who would like to use the device in their lessons including Getting started, Singing Jelly Baby and other features about the Raspberry Pi

3.2.5 POWER SUPPLY:

There are several ways to convert an AC voltage at a all receptacle into the DC voltage required by a microcontroller. Traditionally, this has been done with a transformer and rectifier circuit. There are also switching power supply solutions, however, in applications that involve providing a DC voltage to only the microcontroller and a few other low-current devices, transformer-based or switcher-based power supplies may not be cost effective. The reason is that the transformers in transformer-based solutions, and the inductor/MOSFET/controller in switch-based solutions, are expensive and take up a considerable amount of space. This is especially true in the appliance market, where the cost and size of the components surrounding the power supply may be significantly less than the cost of the power supply alone.

Transformerless power supplies provide a low-cost alternative to transformer-based and switcher-based power supplies. The two basic types of transformerless power supplies are resistive and capacitive .

Transformerless Power Supplies

Transformerless AC power supply theory is not generally taught at the university level, yet the use of such power supplies is prevalent in consumer goods.

Basic Concept

A transformerless power supply typically incorporates:

- Rectification
- Voltage Division
- Regulation
- Filtering
- InrushLimiting

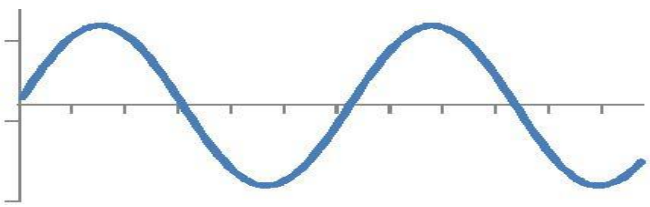
Described simply, the AC input voltage charges up an output filter capacitor. The AC voltage is rectified to ensure that the capacitor is only charged and not discharged by the mains.

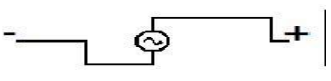
Voltage division ensures that only a small fraction of the input voltage shows up across the output capacitor. Lastly, a Zener diode in parallel with the output capacitor performs basic voltage regulation.

3.2.6 Rectification

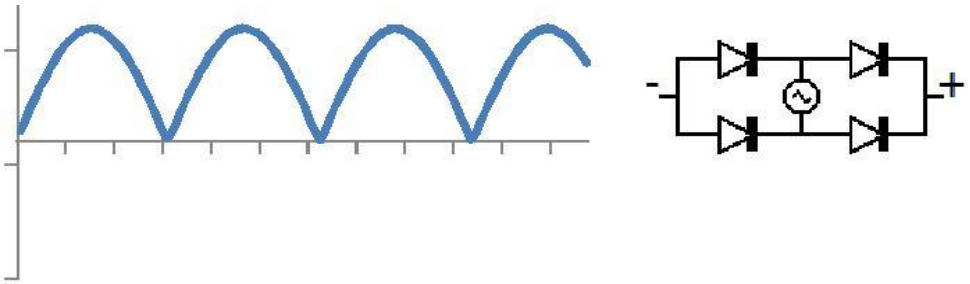
The first step in the process is to rectify the high voltage AC sinusoid such that the remainder of the circuit is only ever exposed to positive voltage. This is achieved by passing the sinusoid through either (1) a full bridge rectifier composed of four diodes, or (2) a single diode that blocks the negative part of the sinusoidal voltage, as shown in Figure 1.

AC Input Voltage





Output with full-wave rectification (four diodes)



Output with half-wave rectification (single diode)

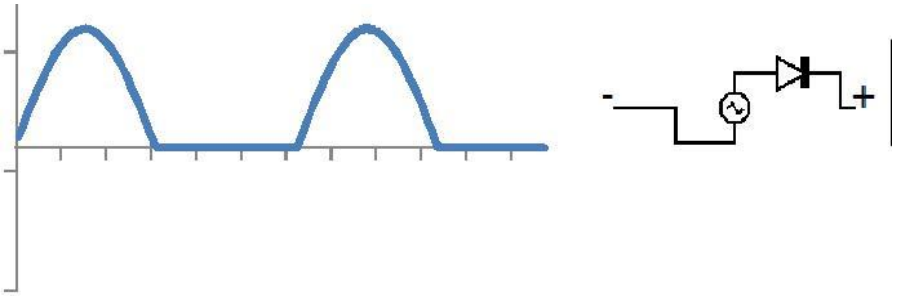


Figure 1 Input and Output Waveforms for Full- and Half-Wave Rectification Stages

Regardless of the configuration, diodes comprise the rectification stage. In the following sections, the rectified output voltage waveform will be referred to simply as V_{in}.

Using a High Voltage to Make a Low Voltage

A well-known method of generating a low voltage from a high voltage is to use a voltage divider circuit, as shown in Figure 2. In textbook examples, the impedances Z_1 and Z_2 are typically resistors, and if only negligible current leaves through V_{out}, then the voltage we can expect at V_{out} is $V_{in} * Z_2 / (Z_1 + Z_2)$.

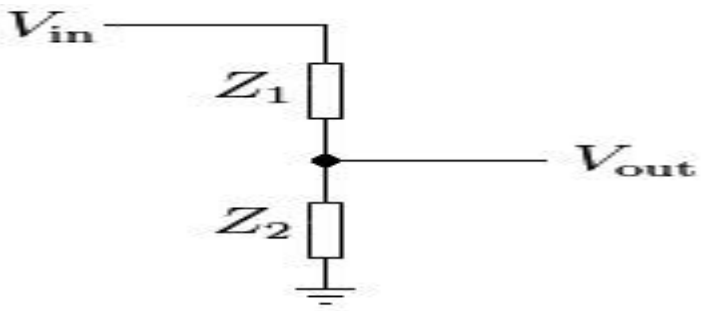


Figure 2 Basic Voltage Divider Circuit

Using resistors for both Z_1 and Z_2 will generally result in a poor power supply design. Good power supplies support a range of output current from V_{out} while holding the output voltage constant. In a resistor-based design, if there is any load current then the voltage drop across Z_1 will increase and V_{out} will correspondingly decrease, which is undesirable. Another issue is that if V_{in} were to decrease, then V_{out} will decrease by the same proportion. Given that V_{in} varies significantly over time (see Figure 1), we know with certainty that V_{out} will also vary over time as well, which is again undesirable.

To improve the performance of the power supply, we can replace the Z_2 resistor with a Zener diode instead, as shown in Figure 3

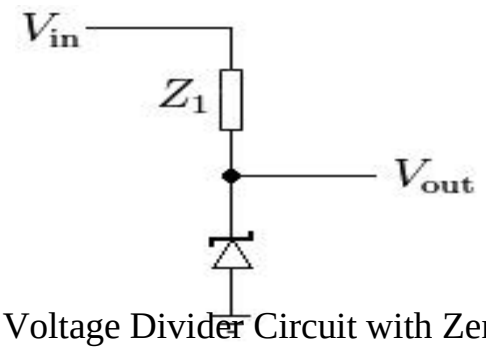


Figure 3 Voltage Divider Circuit with Zener

An ordinary diode will pass current in one direction (indicated by the arrow of the

symbol) but will block current if it tries to flow the other way, preferring instead to allow a reverse voltage to build up across it. The Zener diode has the unusual characteristic that above a specified reverse voltage (the Zener voltage) for the device, it will no longer block current but instead allow current to flow through it in reverse. This reverse current flow occurs only when the reverse voltage across the Zener diode grows high enough to match the Zener voltage rating.

Note that the Zener diode will actively dissipate power when reverse current flows through it because there is a voltage (the Zener voltage) across it. This power is P = I*V.

Using a Zener diode instead of a resistor for Z_2 produces a better power supply which maintains constant output voltage despite changes in input voltage or load current. Regardless of whether 1mA or 30mA is flowing through the Zener diode, its Zener voltage does not change (much). Thus, V_{out} also will not change, even if a load connected to V_{out} draws current (current that would have otherwise passed through the Zener diode).

If the small amount of inherent Zener diode voltage variability is unacceptable, V_{out} can feed into an LDO or DC/DC power supply, which will provide improved output regulation.

3.2.7 Constant Power Loss

The Zener diode introduces a non-obvious drawback common to all transformerless power supplies: constant power consumption regardless of load. The current passing through Z_1 can go one of two places: through the Zener diode or through the load connected to V_{out}. However, the total average current will always match the current through Z_1. For a transformer less supply that can source up to 30mA:

- If the load connected to V_{out} draws very little
  current (or none at all), then all unused current (up to 30mA) flows through Z_2 which dissipates power in the Zener diode.
- If the load connected to V_{out} draws most of the 30mA, then the power dissipation of

the Zener is lower while the power dissipation of the load is higher.

Output (Hold-up) Capacitance

A rectified sinusoidal AC input voltage (as shown in Figure 1) has periods of time where the instantaneous V_{in} has a smaller magnitude than the DC output (Zener) voltage. To prop up the DC output voltage during these periods, a capacitor is added to V_{out}. This capacitor allows V_{out} to "ride through" the periods of small instantaneous AC voltage.

Input Impedance

Z_1 is usually implemented as one of two options. A very simple low cost Z_1 is a resistor; a more efficient option is a capacitor. The size of the Z_1 resistor or capacitor and the Zener voltage together determine how much total output current will be available

Blocking Diode Placement

There are two places where blocking diode(s) for rectification can be placed: before the Zener diode and after the Zener diode.

In general, placing a blocking diode after the Zener ("post-Zener") will prevent the (admittedly small) reverse current flow from the output capacitor through the Zener. The output capacitor generates reverse current flow through the Zener only during portions of the waveform where V_{in} is less than the output capacitor voltage. Inclusion of a Post-Zener diode results in a tradeoff that the output voltage will typically be a diode drop (0.7V) less than the Zener voltage.

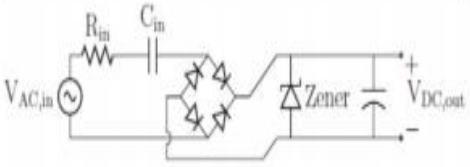
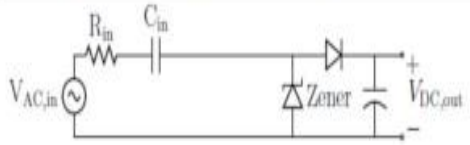
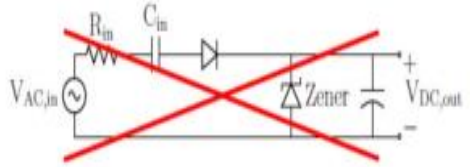
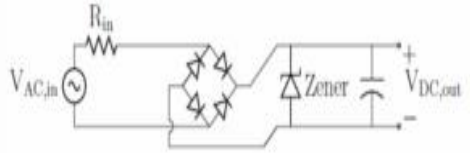
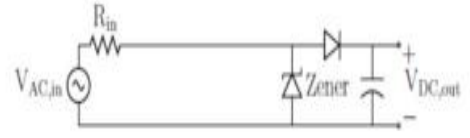
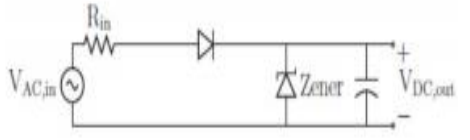
Full Wave Rectified Circuits:

For full-wave rectification to be effective, the rectification must be performed before the Zener diode (that is, the full bridge rectifier must be between the AC source and the Zener diode). This is because the Zener diode will only generate the Zener voltage output whenever a reverse voltage is applied to it. Full wave rectification

ensures that Vin is positive, which allows the Zener voltage to be generated. If full wave rectification were added after the Zener diode (between the Zener and the output capacitor), then the negative portion of the AC waveform would simply result in forward conduction through the Zener, which does not generate a useful output voltage. Therefore, for full-wave rectification, blocking diodes must always be present before the Zener diode ("pre-Zener"). An optional blocking diode may still be placed after the Zener. However, in full wave rectified circuits this is typically not done; the opportunity for the output capacitor to discharge through the Zener in reverse occurs so infrequently that that the leakage is not a concern.

Half Wave Rectified Circuits:

If half wave rectification is used, a single post-Zener blocking diode may be used with no pre-Zener blocking diode. Post-Zener blocking diodes provide greater benefit in a half wave rectified circuit because (as shown in the waveform of Figure 1) at least 50% of the time, V_{in} sits at 0V, which is less than the output voltage, giving plenty of opportunity for capacitor leakage through the Zener. The leakage is even more evident in low voltage Zener diodes (<6V typically) because their current-voltage curve tends to be "softer"—that is, the Zener may start conducting current well before the Zener voltage is reached. In many cases, however, the leakage currents even without a Post-Zener blocking diode is usually low enough that it is not a concern.

 Capacitive Full Wave Rectification (Pre-Zener)	-More efficient than resistive -Physically larger and higher material cost than resistive -Provides twice the output current of the half-wave capacitive -DC output ground not tied to AC neutral; *cannot* drive SCRs/TRIACs directly
 Capacitive Half Wave Rectification (Post-Zener)	-More efficient than resistive -Physically larger and higher material cost than resistive -Provides half the output current of the full-wave capacitive -DC output ground tied to AC neutral; SCRs/TRIACs can be driven directly -No reverse current loss from output capacitor through Zener
 Capacitive Half Wave Rectification (Pre-Zener)	Simply does not work. Output current for capacitive supply is determined by $i_{out} = C_{in}\frac{dv}{dt}$. The diode blocks capacitor C_{in} from discharging, so $dv = 0$ and thus $i_{out} = 0$
 Resistive Full Wave Rectification (Pre-Zener)	-Less efficient than capacitive (significant heat generated by R_{in}) -Physically smaller and lower material cost than capacitive -Provides twice the output current of the half-wave resistive -DC output ground not tied to AC neutral; *cannot* drive SCRs/TRIACs directly -R_{in} power loss is twice as much as resistive half-wave pre-Zener -Slight reverse current loss from output capacitor through Zener (can be mitigated with additional blocking diode)
 Resistive Half Wave Rectification (Post-Zener)	-Less efficient than capacitive (significant heat generated by R_{in}) -Physically smaller and lower material cost than capacitive -Provides half the output current of the full-wave resistive -DC output ground tied to AC neutral; SCR/TRIACs can be driven directly -Output blocking diode drop reduces V_{out} below Zener voltage -R_{in} power loss is twice as much as resistive half-wave pre-Zener -No reverse current loss from output capacitor through Zener
 Resistive Half Wave Rectification (Pre-Zener)	-Less efficient than capacitive (significant heat generated by R_{in}) -Physically smaller and lower material cost than capacitive -Provides half the output current of the full-wave resistive -DC output ground tied to AC neutral; SCR/TRIACs can be driven directly -V_{out} is not reduced by the presence of a blocking diode -R_{in} power loss is half as much as resistive half-wave post-Zener -Slight reverse current loss from output capacitor through Zener (can be mitigated with additional blocking diode)

3.3SOFTWARE REQUIREMENTS

3.3.1OpenCV:

OpenCV (*Open source computer vision*) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.

OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers

3.3.2Frameworks:

- YOLOv5 / YOLOv8 (object detection)
- OpenCV (image/video processing)
- PyTorch or TensorFlow (AI models)
- MediaPipe (optional – hand/pose tracking)

Libraries: ultralytics, torch, opencv-python, numpy, matplotlib, pandas

IDE/Tools: VS Code, Jupyter Notebook, PyCharm

Optional Backend: Flask / FastAPI (for APIs/dashboard)

Optional Deployment: Docker, Google Colab, AWS/GCP

Optional Alerts: Twilio (SMS), Firebase/Pushbullet (push), smtplib (email

Officially launched in 1999 the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects

including real-time ray tracing and 3D display walls.The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described as:

- Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
- Advance vision-based commercial applications by making portable, performance-optimized code available for free – with a license that did not require code to be open or free itself.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008.

The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations.

In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.

On May 2016, Intel signed an agreement to acquire Itseez, a leading developer of OpenCV.

Applications:

openFrameworks running the OpenCV add-on example

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN)[11]

3.3.3 Programming language

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation Wrappers in other languages such as C#, Perl, Ch,Haskell, and Ruby have been developed to encourage adoption by a wider audience.

Since version 3.4, **OpenCV.js** is a JavaScript binding for selected subset of OpenCV functions for the web platform.

All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

Hardware acceleration

If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself.

A CUDA-based GPU interface has been in progress since September 2010.

An OpenCL-based GPU interface has been in progress since October 2012, documentation for version 2.4.13.3 can be found at docs.opencv.org.

OS support

OpenCV runs on the following desktop operating systems: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD. OpenCV runs on the following mobile operating systems: Android, iOS, Maemo, BlackBerry 10. The user can get official releases from SourceForge or take the latest sources from GitHub. OpenCV uses CMake.

COMPARISION BETWEEN OPENCV AND MATLAB

- Open CV is a computer vision library which gives u functions to image and video processing in C/C++.
- It is kind of simplified directshow with many image and video processing function.
- You can build very powerful algorithms which will give you very good efficiency as compared to vision tool box of matlab.
- You can also build commercial applications with it because it has BSD licensing so kind of open source.
- Coming to Matlab ,it is good and very easy but lack effiency and speed.
- Since matlab is a scripting language unlike Opencv which gets complied.
- It is difficult to make video conferencing or any other video processing efficient since you can see any encoder is always built using c/c++ rather I would say c because it is more close to assembly.
- so if you are trying to just do some research and analysis then go for matlab

- But if you want to make a commercial software then go for opencv as you can include other c libraries if needed.
- But you will have little hard time learning opencv if you are a beginner in c/C++

Advantages of OpenCV over MATLAB

Speed: Matlab is built on Java, and Java is built upon C. So when you run a Matlab program, your computer is busy trying to interpret all that Matlab code. Then it turns it into Java, and then finally executes the code. OpenCV, on the other hand, is basically a library of functions written in C/C++. You are closer to directly provide machine language code to the computer to get executed. So ultimately you get more image processing done for your computers processing cycles, and not more interpreting. As a result of this, programs written in OpenCV run much faster than similar programs written in Matlab. So, conclusion? OpenCV is damn fast when it comes to speed of execution. For example, we might write a small program to detect peoples smiles in a sequence of video frames. In Matlab, we would typically get 3-4 frames analysed per second. In OpenCV, we would get at least 30 frames per second, resulting in real-time detection.

Resources needed: Due to the high level nature of Matlab, it uses a lot of your systems resources. And I mean A LOT! Matlab code requires over a gig of RAM to run through video. In comparison, typical OpenCV programs only require ~70mb of RAM to run in real-time. The difference as you can easily see is HUGE!

Cost: List price for the base (no toolboxes) MATLAB (commercial, single user License) is around USD 2150. OpenCV (BSD license) is free! Now, how do you beat that? Huh? huh? huh?

Portability: MATLAB and OpenCV run equally well on Windows, Linux and MacOS. However, when it comes to OpenCV, any device that can run C, can, in all probability, run OpenCV.

Despite all these amazing features, OpenCV does lose out over MATLAB on some po

Ease of use: Matlab is a relatively easy language to get to grips with. Matlab is a pretty high-level scripting language, meaning that you don't have to worry about libraries, declaring variables, memory management or other lower-level programming issues. As such, it can be very easy to throw together some code to prototype your image processing idea.

Memory Management: OpenCV is based on C. As such, every time you allocate a chunk of memory you will have to release it again. If you have a loop in your code where you allocate a chunk of memory in that loop and forget release it afterwards, you will get what is called a "leak". This is where the program will use a growing amount of memory until it crashes from no remaining memory. Due to the high-level nature of Matlab, it is "smart" enough to automatically allocate and release memory in the background.

Development Environment: Matlab comes with its own development environment. For OpenCV, there is no particular IDE that you have to use. Instead, you have a choice of any C programming IDE depending on whether you are using Windows, Linux, or OS X. For Windows, Microsoft Visual Studio or NetBeans is the typical IDE used for OpenCV. In Linux, its Eclipse or NetBeans, and in OSX, we use Apple's Xcode.

3.3.4 IoT PLATFORM

An IoT platform is essential for managing IoT devices, handling data ingestion, facilitating device communication, and ensuring security. These platforms provide centralized control and management capabilities, enabling seamless integration of diverse IoT devices into a unified system. Popular IoT platforms include AWS IoT, Google Cloud IoT, Microsoft Azure IoT, Blynk IoT, as well as open-source platforms like Eclipse IoT, offering flexibility and scalability t accommodate various project needs.

1. ThingSpeak

Best for: Real-time data logging and basic analytics

- ThingSpeak is an open-source IoT platform that allows devices to send data over HTTP or MQTT.
- Ideal for storing detection logs (e.g., when and where smartphones were detected).

LEARNING TOOLS

Purpose: Train and deploy AI models for smartphone detection

- **Google Colab / Jupyter Notebook**: Free platforms to write and execute Python code with GPU support. Ideal for model training and testing.
- **PyTorch / TensorFlow**: Deep learning libraries to build, train, and deploy object detection models like YOLO.
- **YOLOv5 / YOLOv8**: Pretrained object detection models that can detect smartphones in real-time video feeds.
- **OpenCV**: For image and video processing, integrating camera feeds, and drawing detection boxes.

REAL-TIME DATA PROCESSING

Purpose: Analyze video frames continuously to detect smartphones in real-time

- **OpenCV with Python**: Used to capture live video from webcam/CCTV, apply detection model, and process frames instantly.
- **Multithreading / multiprocessing**: For handling camera feeds, AI inference, and alert systems without lag.
- **Edge devices (Raspberry Pi, Jetson Nano)**: Can process data locally with lightweight models for low-latency detection.

3.3.5 WEB DEVELOPMENT FRAMEWORK

Purpose: Create dashboards, logs, and user interfaces for monitoring

- **Flask**: Lightweight Python web framework to build REST APIs and simple web dashboards for data visualization and alerts.
- **FastAPI**: High-performance Python framework ideal for building real-time APIs .

3.3.6 DATABASE MANAGEMENT SYSTEM (DBMS):

Purpose: Store detection logs, timestamps, user activity, and alert history

- **SQLite**: Lightweight, file-based database for local projects (no setup needed).
- **MySQL / PostgreSQL**: Powerful relational databases for multi-user systems and scalable storage.
- **Firebase Realtime Database**: Cloud-based NoSQL database suitable for mobile/web syncing and real-time data storage.
- **MongoDB**: NoSQL document-based database – useful for storing JSON-like detection data and logs.

project.

3.3.7 INTEGRATION AND API'S:

The APIs are essential to **exchange data** between modules like detection logic, dashboards, mobile apps, and cloud platforms.

- **REST APIs (Flask / FastAPI)**:
- Used to send smartphone detection data from AI system to web/mobile dashboard.
- Example: When a phone is detected, a POST request sends the info (time, location, image) to a web server.
- **Firebase API**:
- Used for real-time syncing of data between device and mobile/web app.
- Supports user authentication, cloud storage, and push notifications.
- **ThingSpeak / Blynk APIs**:
- Used for IoT data logging, live dashboards, and triggering remote alerts.
- API keys ensure secure communication with these platforms.
- **Camera/Media APIs (OpenCV)**:
    - Access video streams for live frame processing.

2. Integration

- Integration involves **connecting all modules** into a working system:
- **AI Model ↔ OpenCV**:
    - Integrate YOLOv5/YOLOv8 with OpenCV to process video frames in real-time.
- **OpenCV ↔ Flask API**:
    - Send detection results (e.g., "phone detected at 10:23 AM") via API to database or web dashboard.
- **Flask ↔ DBMS (SQLite/MySQL/Firebase)**:
    - Store data like timestamps, detected objects, user info.
- **Flask ↔ Notification APIs (Twilio/Firebase)**:
    - On detection, send alert to teacher's mobile via SMS or push notification.
- **IoT Devices ↔ Cloud APIs (ThingSpeak/Blynk)**:
    - Log sensor data (if present) and trigger real-time feedback like buzzers or lights in the classroom.
- **Web Dashboard ↔ REST API**:
    - Fetch and display detection history, alerts, and real-time status ecosystem.

3.3.7 ALERT NOTIFICATION:

The alert notification system is a key component of the AI-Based Smartphone Detection in Classroom project. It provides **real-time, physical alerts** inside the classroom when unauthorized smartphone usage is detected by the AI model.
This system uses a **5V active buzzer** connected to a **microcontroller or edge device** (such as Raspberry Pi or Jetson Nano). When the AI model processes the live video feed and identifies a smartphone, it immediately triggers a signal through a **GPIO (General Purpose Input/Output) pin**. The buzzer is then activated, producing an audible beep to **alert the teacher and discourage students from using smartphones** during class hours. The buzzer can be programmed to remain active for a specific time (e.g., 2–3 seconds) or until no further smartphone activity is detected in the video feed. A **transistor-based switching mechanism** is used if the buzzer requires higher current, ensuring stable operation and protection of the control device.

Smartphone Detection:

The AI model (e.g., YOLOv5/YOLOv8) processes the camera feed using OpenCV. When a smartphone is detected in a video frame, it raises a detection flag.

Signal Sent to Microcontroller:

The Python script running on your main device (Raspberry Pi, Jetson Nano, or a PC with GPIO support) sends a digital output signal (HIGH) to a GPIO pin connected to the buzzer.

Buzzer Activation:

The buzzer is wired through a transistor or relay (if required) and gets activated when the signal is HIGH, producing a beep sound.

.

CHAPTER 4

PROPOSED MODEL

4.1 INTRODUCTION

The proposed model for the **AI-Based Smartphone Detection in the Classroom** utilizes a combination of computer vision, artificial intelligence, and IoT to detect and monitor smartphone usage during class hours. The system begins by capturing live video footage from a **camera module or CCTV** installed in the classroom. These cameras are strategically placed to cover a wide area of the classroom, ensuring that all students are visible. The captured video feed is then processed in real-time using an **AI object detection model** such as **YOLOv5 or YOLOv8**, which is trained to recognize smartphones. This model runs on a processing unit like a **Raspberry Pi**, **Jetson Nano**, or a dedicated PC, depending on the processing power required for real-time detection.

Once the AI model detects a smartphone in the classroom, it triggers an **event** that includes logging the timestamp and capturing an image of the detection. This event activates an **alert system**, typically a **buzzer** connected to the system via GPIO pins. The buzzer emits a sound for 2–3 seconds, notifying both the teacher and students in the classroom of the detected smartphone usage. For added functionality, visual alerts can also be triggered using **LED lights or an LCD display** to complement the buzzer.

To further enhance the system, **API integration** is employed. A **Flask** or **FastAPI** server is used to create **REST APIs** that allow communication between the AI model, the database, and the user interface. If needed, the system can send real-time alerts to faculty or administrators via services like **Twilio** (SMS, voice calls) or **Firebase** (push notifications). These notifications can be customized and delivered to relevant parties to prompt immediate action. The system also stores detection data in a **local database** or **cloud-based service like Firebase**, which includes event logs, timestamps, and

captured images for later review and analysis.

The **user interface** is built as a **web dashboard** that allows teachers or administrators to monitor real-time detection, view historical logs, and control system settings such as detection sensitivity or alert configurations. This dashboard can also display the live camera feed and show any current alerts or notifications. By combining AI-powered smartphone detection with real-time notifications and robust data logging, this system ensures that classroom management remains effective and efficient, promoting a focused learning environment.

The **AI-Based Smartphone Detection in the Classroom** project aims to implement a comprehensive, automated monitoring system that detects smartphone usage among students during class hours. The process begins with the **data acquisition** layer, where a **camera module or CCTV** system is installed in the classroom. These cameras are placed strategically around the room to provide wide coverage, ensuring that the entire class can be monitored effectively. The video feed from these cameras is sent to a processing unit, such as a **Raspberry Pi**, **Jetson Nano**, or a dedicated PC, which is responsible for running the AI algorithms.

In the **AI processing layer**, the system utilizes a pre-trained **object detection model** such as **YOLOv5 or YOLOv8** to analyze the video frames. The AI model is trained to recognize specific objects, in this case, smartphones, within the classroom environment. As the video feed is processed, the AI model continuously scans each frame to identify any smartphones that are present. When a smartphone is detected, the model triggers an event that logs the detection, including the time of detection and a snapshot of the classroom. This event is essential for keeping track of when and where smartphones are being used, creating a valuable dataset for the teacher or administrator to review.

Once a smartphone is detected, the system enters the **alert notification stage**. The event triggers a **buzzer system** connected to a microcontroller (such as a Raspberry Pi

or Arduino) via a **GPIO pin**. This buzzer emits a sound that lasts for 2–3 seconds, notifying the class that a smartphone has been detected. This provides immediate feedback to the students and alerts them that smartphone usage is prohibited during class time. For added clarity, visual alerts can be implemented using **LED lights** or an **LCD display** that shows the status of the detection, such as "Smartphone Detected: Class in Progress" or other custom messages.

To expand the functionality of the system, **integration with external services** is used. The system can be enhanced with a **Flask** or **FastAPI** backend, which provides **REST APIs** to facilitate communication between the different components of the system. This allows the AI system to interact seamlessly with the **database** where detection logs and events are stored. Additionally, the backend can be configured to send alerts to teachers, administrators, or even students through communication platforms like **Twilio** or **Firebase**. These platforms allow for sending real-time **SMS, voice calls, or push notifications** to ensure that the relevant parties are notified as soon as a smartphone is detected. For example, a teacher could receive an **SMS alert** every time a smartphone is detected, allowing them to take action immediately.

The **database layer** is crucial for data management. All detection events, including timestamps, images of the classroom, and the type of object detected (smartphone), are logged in a database such as **Firebase**, **MySQL**, or **SQLite**. This data is stored for future analysis and auditing, allowing the teacher or administrator to review historical data and take necessary actions based on patterns or trends. For example, if a student repeatedly uses their phone in class, the teacher could use the database to check the logs and address the issue accordingly.

4.2 Monitoring and Integration

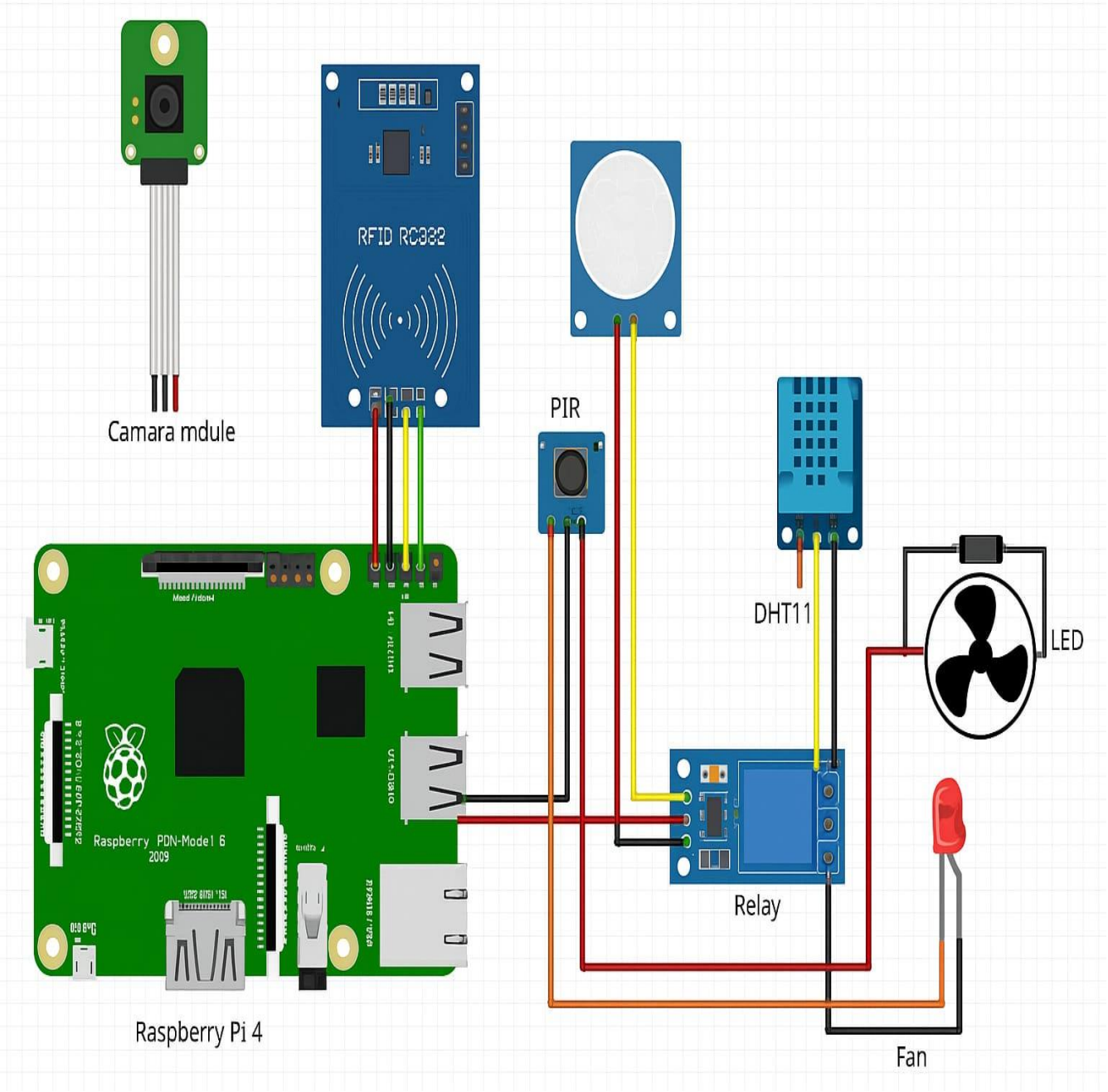
The AI-based smartphone detection system is designed to monitor classroom environments in real time, ensuring students remain focused and free from digital distractions. The monitoring process involves the continuous capture of video or sensor data from strategically placed surveillance cameras or IoT devices within the classroom. These inputs are processed using advanced AI models, such as CNN, YOLO, or SSD, which are trained to detect the presence of smartphones based on their physical features and usage behavior. Once a smartphone is detected, the system automatically triggers alerts to the faculty or administrative dashboard, allowing for timely intervention. The integration of this system with existing classroom infrastructure is seamless and non-intrusive. It can be incorporated into existing CCTV systems or linked with school management software for centralized control and reporting. Furthermore, the integration ensures data privacy by processing images locally or anonymizing visual data where necessary. The system may also be enhanced with network-based detection (e.g., Bluetooth or Wi-Fi signal scanning) to identify active mobile devices even when they are hidden from view. Overall, the real-time monitoring and smart integration of AI-driven detection not only promote discipline but also help create a more focused and distraction-free learning environment.

.

To ensure **ethical compliance**, privacy-preserving methods such as on-device edge processing, data anonymization, and access control mechanisms are incorporated. The system only detects objects of interest (smartphones) and avoids facial recognition or storing personal information, addressing concerns about student privacy. Overall, the AI-based smartphone detection system offers a scalable, automated, and intelligent solution that integrates seamlessly with school infrastructure, helping institutions maintain classroom discipline and improve academic engagement without requiring manual supervision

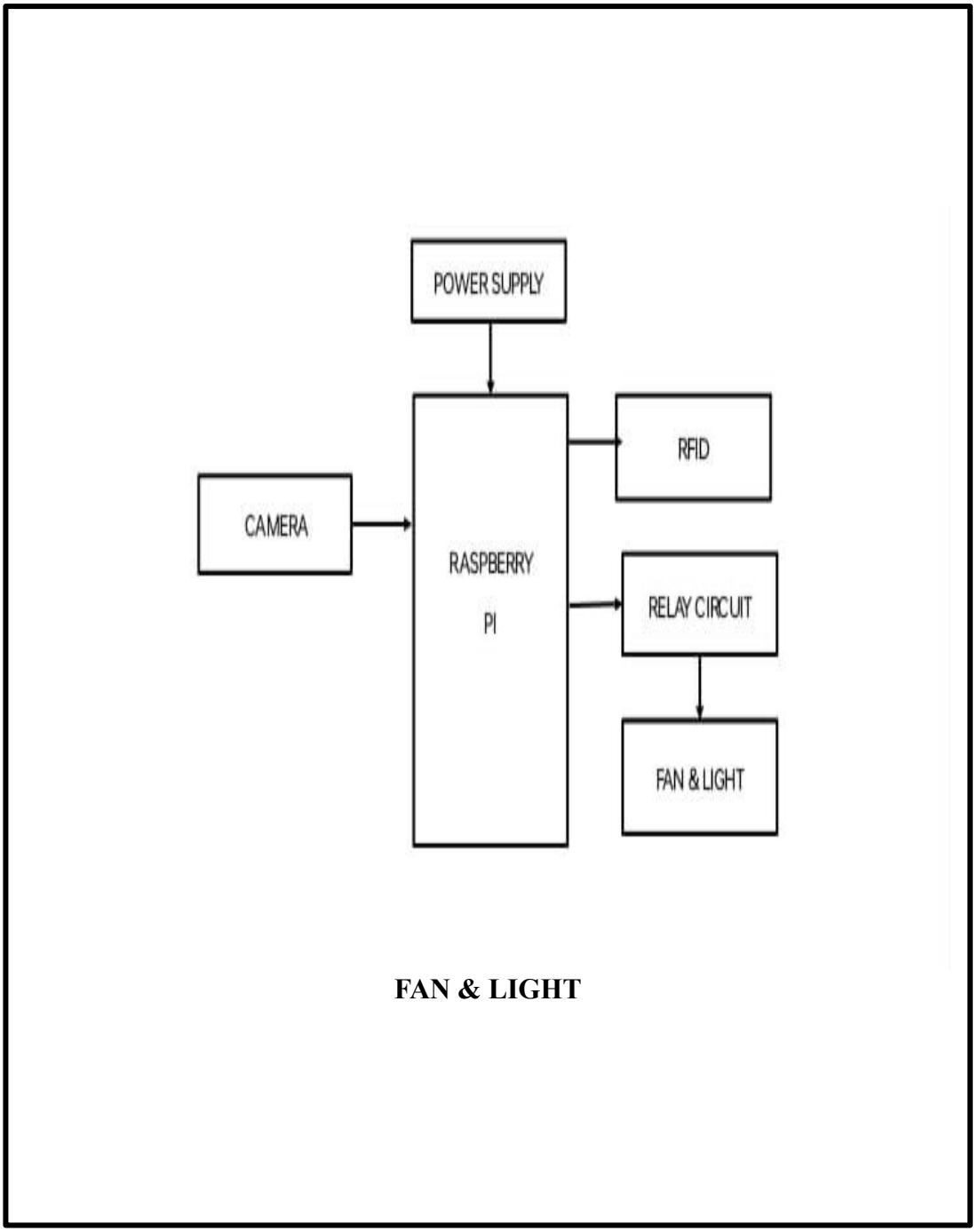
4.3 CIRCUIT DIAGRAM

Figure 4.1 Circuit Diagram



4.4 BLOCK DIAGRAM

Figure 4.2 System Architecture



4.5 YOLO (You Only Look Once) Algorithm

In the **AI-Based Smartphone Detection in Classroom** project, the primary algorithm used is **YOLO (You Only Look Once)**, an advanced deep learning-based object detection model that is well-suited for real-time applications. YOLO works by dividing an image into a grid and predicting bounding boxes around potential objects, such as smartphones, within those grid regions. This algorithm is trained using a labeled dataset containing images of classrooms with annotated smartphones. Once trained, YOLO can efficiently detect smartphones in each video frame of the classroom in real-time. YOLO's fast processing speed makes it an ideal choice for this project, as it can handle video feeds and identify objects quickly enough to trigger alerts.

The detection process is supported by **Convolutional Neural Networks (CNNs)**, which are integral to YOLO. CNNs are used for feature extraction, automatically learning important visual features like shapes and edges in images. These features are then used to classify objects (in this case, smartphones). The YOLO model integrates CNNs to extract features from each frame and classify them, ensuring high accuracy in detecting smartphones even in cluttered environments like classrooms.

After detecting potential smartphones, the system uses **Non-Maximum Suppression (NMS)** to filter out duplicate detections. Since multiple bounding boxes can overlap when a single object is detected, NMS compares the confidence scores of each prediction and removes the boxes with the highest overlap, keeping only the most accurate one. This ensures that only one alert is triggered for each smartphone detected.

Real-time processing is essential for this project, and the system processes

each video frame independently, making smartphone detection possible in a continuous classroom setting. As soon as a smartphone is detected in a frame, the system generates an event that triggers the appropriate alert—typically a **buzzer sound** in the classroom. The system can also send **real-time notifications** to teachers or administrators through services like **Twilio** or **Firebase**, alerting them of the smartphone usage.

In summary, the project leverages the power of **YOLO** for real-time object detection, **CNNs** for feature extraction, **NMS** for handling overlapping detections, and an event-driven system to trigger alerts. This combination of algorithms ensures efficient and effective smartphone detection, providing real-time alerts that maintain classroom discipline.

In the **AI-Based Smartphone Detection in Classroom** project, the primary algorithm employed is **YOLO (You Only Look Once)**, a state-of-the-art deep learning model for real-time object detection. YOLO's approach to object detection is unique because it performs the detection in a single pass, dividing the image into a grid and predicting bounding boxes along with class probabilities for the objects present in each grid cell. The advantage of YOLO is its speed and efficiency, making it ideal for real-time applications like classroom monitoring. In this system, YOLO is specifically trained to recognize smartphones, utilizing a labeled dataset where images of smartphones in classrooms are annotated. Once the model is trained, it can detect smartphones in live video frames streamed from cameras placed around the classroom.

The core of the detection process relies heavily on **Convolutional Neural Networks (CNNs)**, which are the backbone of YOLO for feature extraction. CNNs automatically learn spatial hierarchies of features in the input images—such as edges, textures, and shapes—which are essential for detecting objects accurately. The CNN’s ability to extract high-level features allows YOLO to detect even

partially obscured or small smartphones, ensuring that the detection system is robust enough to handle a variety of classroom environments and smartphone placements.

When multiple objects are detected in the same region of an image, the algorithm can output multiple bounding boxes, which might overlap. This is where **Non-Maximum Suppression (NMS)** plays a crucial role. NMS filters out redundant bounding boxes by keeping only the one with the highest confidence score, effectively removing duplicate detections of the same object. This ensures that each smartphone is detected only once and eliminates unnecessary alerts, making the system more reliable and accurate.

As the system processes each frame in real-time, it monitors the classroom for smartphone usage continuously. Whenever a smartphone is detected in the classroom, an event is triggered, which activates the **alert system**. This alert can be in the form of a **buzzer sound**, which notifies both the teacher and the students in the classroom. Additionally, the system can send notifications to the teacher or administrative personnel through external communication services like **Twilio** or **Firebase**, providing them with real-time updates about the detection event. These notifications can be in the form of **SMS, voice calls**, or **push notifications**, ensuring that the relevant individuals are informed regardless of their physical location in the building.

To enhance the functionality of the system, **data logging** plays an important role. Each detection event is logged in a **database**, capturing key details such as the time of detection, the location (camera ID), and even an image snapshot of the classroom at the moment of detection. This data can be used for further analysis, helping teachers or administrators to identify patterns of smartphone usage over time, such as identifying specific students or times when smartphones are

frequently used. Additionally, the logged data could help in fine-tuning the system to improve detection accuracy or adjust alert parameters.

4.5.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are the backbone of the visual recognition capability in the AI-Based Smartphone Detection in Classroom project. They are specially designed neural networks that excel at analyzing and interpreting pixel data from images and videos. In this project, CNNs are trained to recognize the physical presence of smartphones in classroom environments, whether the device is held in hand, lying on a desk, or partially hidden. The key strength of CNNs lies in their ability to automatically extract complex features from raw images without the need for manual programming or hand-crafted filters.

A typical CNN consists of several layers: **convolutional layers**, which apply learnable filters to detect visual patterns (such as phone edges, screen shapes, or brand logos); **activation layers**, like ReLU, which introduce non-linearity; **pooling layers**, which downsample feature maps to reduce computation while preserving important features; and **fully connected layers**, which interpret the extracted features and classify the image into categories — in this case, "smartphone detected" or "no smartphone." The model is trained using a large, labeled dataset of classroom images with and without smartphones, and it learns to distinguish the relevant features that define smartphone usage.

During real-time classroom monitoring, the CNN processes video frames from surveillance cameras. As the system scans each frame, it highlights the regions where a smartphone is present and flags it. This process is fast and efficient, allowing for live detection with minimal delay. CNNs are also highly adaptable — with techniques like data augmentation and transfer learning, the model can be fine-tuned to work across different classroom setups, lighting conditions, and even recognize new phone models over time.

CHAPTER 5

EXPERIMENTAL ANALYSIS AND RESULTS

5.1 SOURCE CODE:

```
import cv2

import numpy as np

import time

import RPi.GPIO as GPIO

import Adafruit_DHT

GPIO.setmode(GPIO.BCM)

FAN_PIN = 17

LIGHT_PIN = 27

PIR_PIN = 22


GPIO.setup(FAN_PIN, GPIO.OUT)

GPIO.setup(LIGHT_PIN, GPIO.OUT)

GPIO.setup(PIR_PIN, GPIO.IN)

DHT_SENSOR = Adafruit_DHT.DHT11

DHT_PIN = 4


net = cv2.dnn.readNet("yolo.weights", "yolo.cfg")

layer_names = net.getLayerNames()
```

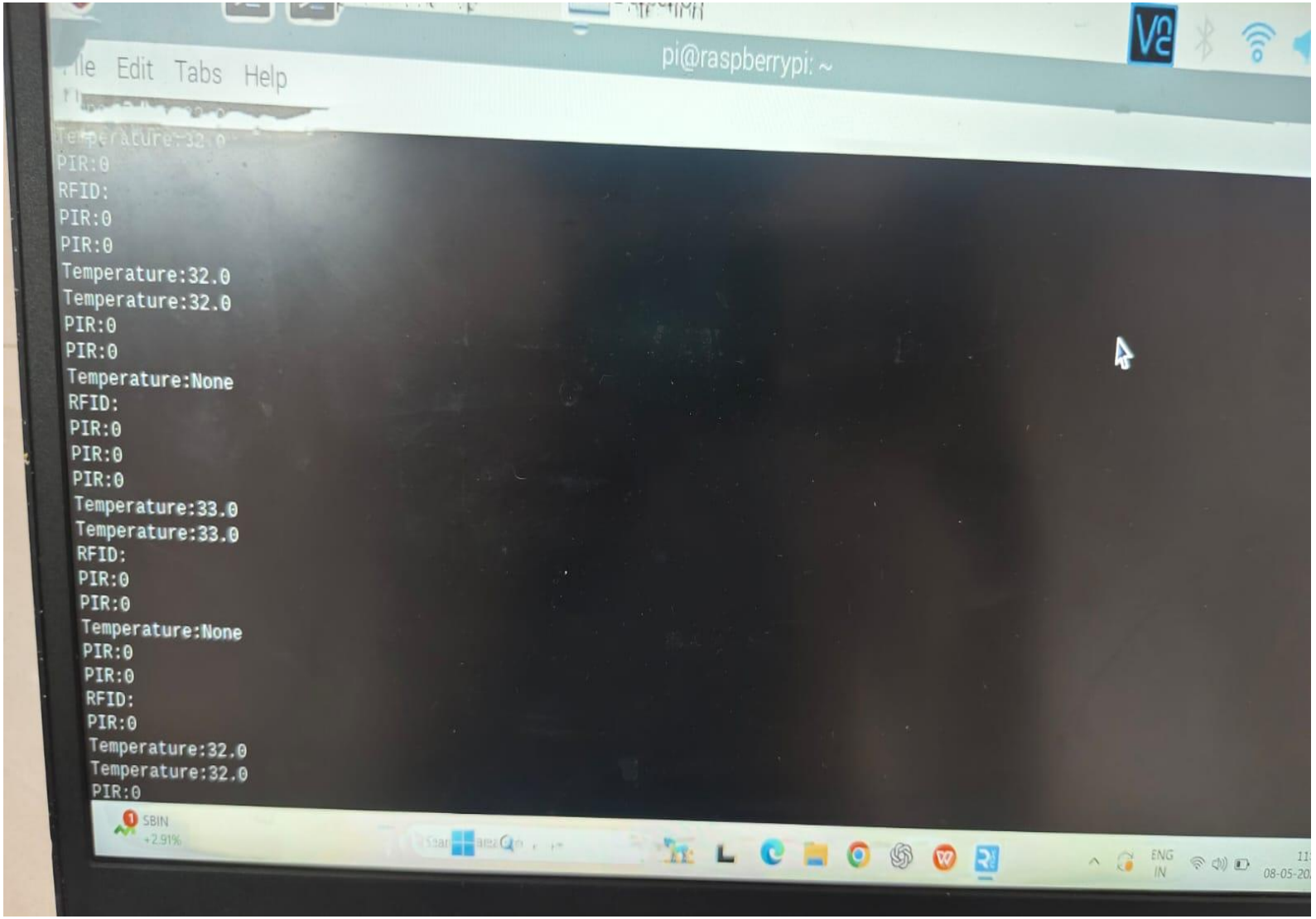
```
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers().flatten()]

with open("coco.names", "r") as f:

classes = [line.strip() for line in f.readlines()]

cellphone_class_index = classes.index("cell phone")

def detect_cellphone(frame):
    height, width = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, scalefactor=1/255, size=(416, 416),
swapRB=True)
    net.setInput(blob)
    outputs = net.forward(output_layers)

    for output in outputs:
        for detection in output:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]

            if class_id == cellphone_class_index and confidence > 0.5:
```

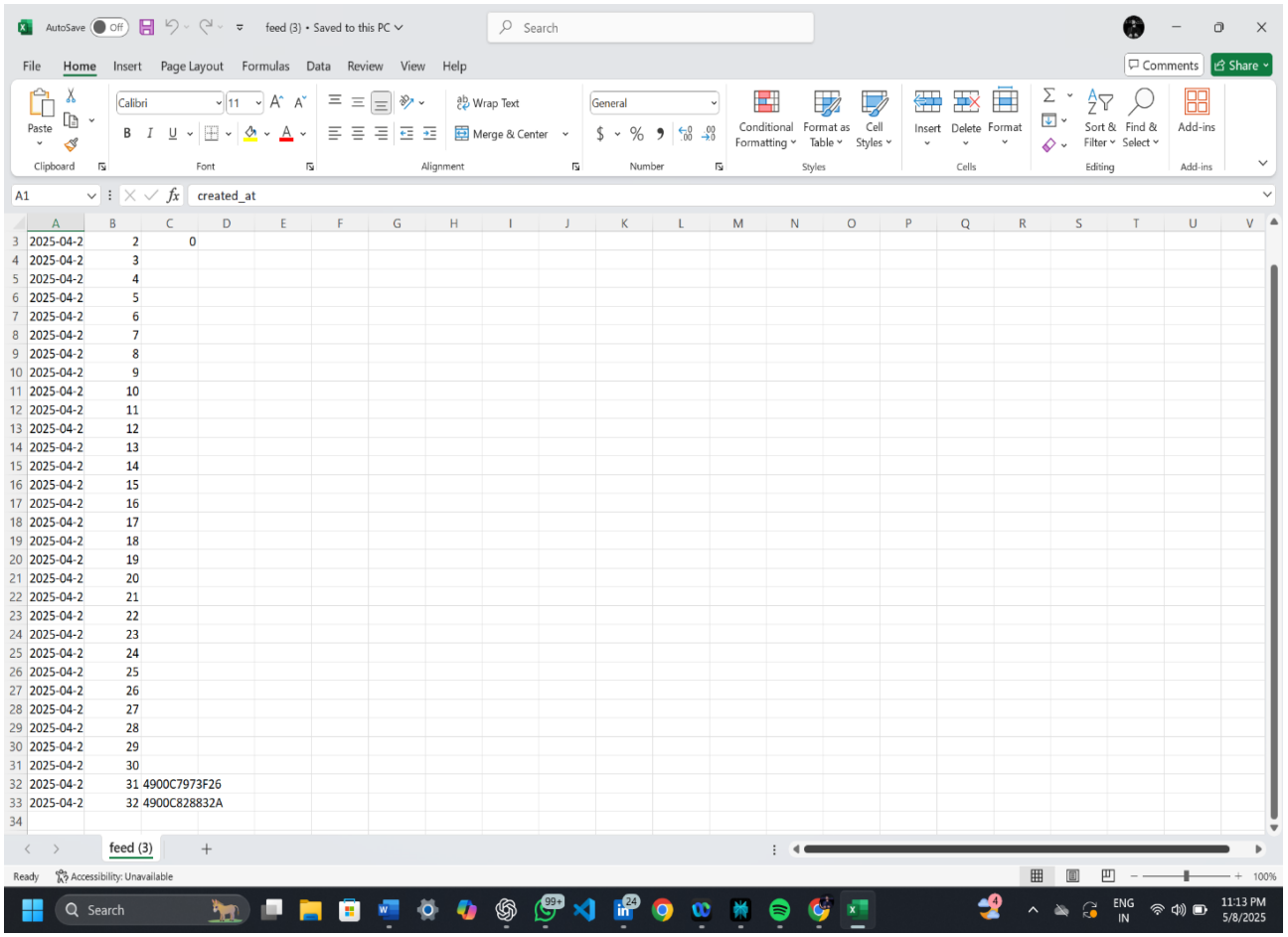
```
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)

            cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,0), 2)
cv2.putText(frame, "Cellphone", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,255,0), 2)
            return True
    return False

def check_temperature():
    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
    if temperature:
        print(f"Temperature: {temperature:.1f}°C")
        return temperature
    return None

def check_motion():
    return GPIO.input(PIR_PIN) == GPIO.HIGH
```

```
def control_devices(temperature, motion):

    if temperature and temperature > 38:
        GPIO.output(FAN_PIN, GPIO.HIGH)
        print("Fan ON (high temp)")
    else:
        GPIO.output(FAN_PIN, GPIO.LOW)



    if motion:
        GPIO.output(LIGHT_PIN, GPIO.HIGH)
        print("Light ON (motion detected)")
    else:
        GPIO.output(LIGHT_PIN, GPIO.LOW)
def main():
    cap = cv2.VideoCapture(0)
    try:
        while True:
            ret, frame = cap.read()
            if not ret:
```

```
            break
        cellphone_detected = detect_cellphone(frame)

temperature = check_temperature()
        motion = check_motion()
        control_devices(temperature, motion)

        if cellphone_detected:
            print("Cellphone detected!")

        cv2.imshow("Raspberry Pi Camera", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    finally:
        cap.release()
        cv2.destroyAllWindows()
        GPIO.cleanup()

if _name_ == "_main_":
    main()
```

5.2 OUTPUT


File Edit Tabs Help
pi@raspberrypi: ~
PIR:0
RFID:
PIR:0
PIR:0
Temperature:32.0
Temperature:32.0
PIR:0
PIR:0
Temperature:None
RFID:
PIR:0
PIR:0
PIR:0
Temperature:33.0
Temperature:33.0
RFID:
PIR:0
PIR:0
Temperature:None
PIR:0
PIR:0
RFID:
PIR:0
Temperature:32.0
Temperature:32.0
PIR:0
SBIN
+2.91%
ENG
IN



	A	B	C
3	2025-04-2	2	0
4	2025-04-2	3	
5	2025-04-2	4	
6	2025-04-2	5	
7	2025-04-2	6	
8	2025-04-2	7	
9	2025-04-2	8	
10	2025-04-2	9	
11	2025-04-2	10	
12	2025-04-2	11	
13	2025-04-2	12	
14	2025-04-2	13	
15	2025-04-2	14	
16	2025-04-2	15	
17	2025-04-2	16	
18	2025-04-2	17	
19	2025-04-2	18	
20	2025-04-2	19	
21	2025-04-2	20	
22	2025-04-2	21	
23	2025-04-2	22	
24	2025-04-2	23	
25	2025-04-2	24	
26	2025-04-2	25	
27	2025-04-2	26	
28	2025-04-2	27	
29	2025-04-2	28	
30	2025-04-2	29	
31	2025-04-2	30	
32	2025-04-2	31	4900C7973F26
33	2025-04-2	32	4900C828832A
34			

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The AI-Based Smartphone Detection in Classroom project provides an innovative and efficient solution to address the misuse of mobile phones during academic sessions. By using advanced object detection techniques like YOLO and real-time video analysis, the system is capable of accurately identifying smartphones in a classroom setting and generating immediate alerts through mechanisms such as buzzer systems and notification APIs. This reduces manual monitoring efforts for teachers and maintains a disciplined learning environment. The integration of AI, IoT, and automation in this project showcases how emerging technologies can be effectively applied in educational settings to enhance classroom management and focus.

6.2 FUTURE WORK

The aspects In the future, this system can be enhanced with additional features such as facial recognition to identify specific students using smartphones, allowing for personalized monitoring and behavioral tracking. Integration with cloud-based platforms can enable long-term data storage and analysis, supporting trend evaluation and administrative reporting. The alert mechanism can be further extended to include mobile app notifications or emails to parents or school authorities. Moreover, deploying the system on embedded edge devices like Raspberry Pi with camera modules can reduce infrastructure costs and make the solution scalable and feasible for low-resource schools. Continuous improvements in AI model accuracy and real-time processing can further refine the system's efficiency and expand its applications beyond classrooms into exam halls or other restricted zones.

The AI-Based Smartphone Detection system has vast potential for future development and real-world applications beyond its current scope. One of the key advancements could be the integration of **facial recognition** to identify students who repeatedly violate smartphone policies. This could allow schools to maintain individual behavior records, enabling teachers and administrators to provide tailored disciplinary actions or counseling.

Another major area for enhancement is **cloud integration**. By storing detection logs, video clips, and alert history in the cloud, educational institutions can generate long-term reports and analytics to study behavioral trends, detect patterns in device usage, and evaluate the effectiveness of their policies. This data could also help in conducting parent-teacher meetings with evidence-based discussions.

Furthermore, the system could be extended to support **multi-classroom and campus-wide monitoring** by implementing a centralized dashboard. This dashboard can show live detection feeds and alerts from all classrooms, helping school authorities monitor student behavior in real-time across the entire institution.

Additionally, the alert system can be expanded to include **mobile push notifications, email alerts, or app-based dashboards** for both staff and parents. These can help in instant communication and better transparency.

On the hardware side, the system can be optimized for deployment on **edge AI devices** like the NVIDIA Jetson Nano or Raspberry Pi with AI accelerators, making it more cost-effective and scalable for schools with limited budgets. It can also be made energy-efficient for round-the-clock operation without significant power consumption.

The system can also incorporate **context-aware intelligence**, using AI to

distinguish between permitted and unpermitted smartphone usage, such as detecting whether the phone is being used for learning or personal activities.

Finally, the model can be adapted to detect **other prohibited items or behaviors** such as cheating during exams, use of earphones, or even drowsiness and inattentiveness, turning it into a comprehensive smart classroom surveillance tool.

These future advancements could transform the current system into a holistic classroom management and student monitoring solution powered by AI and IoT.

Beyond its current capabilities, the smartphone detection system can evolve into a **smart surveillance suite** by incorporating **multi-modal sensor fusion**. For instance, integrating sound sensors or microphones could allow the system to detect ringtone sounds or whispered conversations, thereby enhancing the detection of phone misuse even when the device is not visually visible. Similarly, integrating **infrared or thermal cameras** could detect heat signatures of electronic devices, offering functionality even in low-light or dark environments such as during exams.

A major innovation could involve the development of a **dedicated mobile application** for teachers and administrators. This app could offer live feeds, push alerts, student profiles with behavioral analytics, and the ability to remotely trigger alarms or notifications, giving educators more flexibility and control.

From a **policy and education** perspective, the system's data analytics can be used to inform school policy-making. By analyzing patterns of smartphone use, schools can adjust teaching schedules, introduce digital detox hours, or launch awareness campaigns about the impact of digital distractions. Moreover, predictive analytics models can be built to **forecast high-risk periods or classrooms** prone to frequent phone use, enabling proactive intervention.

The system can also be extended into **university environments or corporate**

training centers, where maintaining attention during seminars is crucial. Custom adaptations of the model can support detection of smartwatches, tablets, or other gadgets that may distract learners.

Another interesting direction is **ethical AI integration**—implementing privacy-preserving AI methods such as **federated learning**, so that the system can learn and adapt across multiple classrooms or schools without transmitting sensitive video data to central servers. This protects student identity and adheres to privacy regulations like GDPR or India's Personal Data Protection Bill.

In terms of scalability and commercialization, the project could evolve into a **product-as-a-service (PaaS)** model, where schools subscribe to a managed system including hardware, software, cloud storage, and analytics dashboard support. Partnerships could be formed with edtech companies or school management software vendors to offer bundled solutions.

Finally, the system can be enhanced with **voice recognition** and **natural language processing (NLP)** to detect students trying to command voice assistants like Siri or Google Assistant during lectures—thereby covering both visual and audio channels of smartphone misuse.

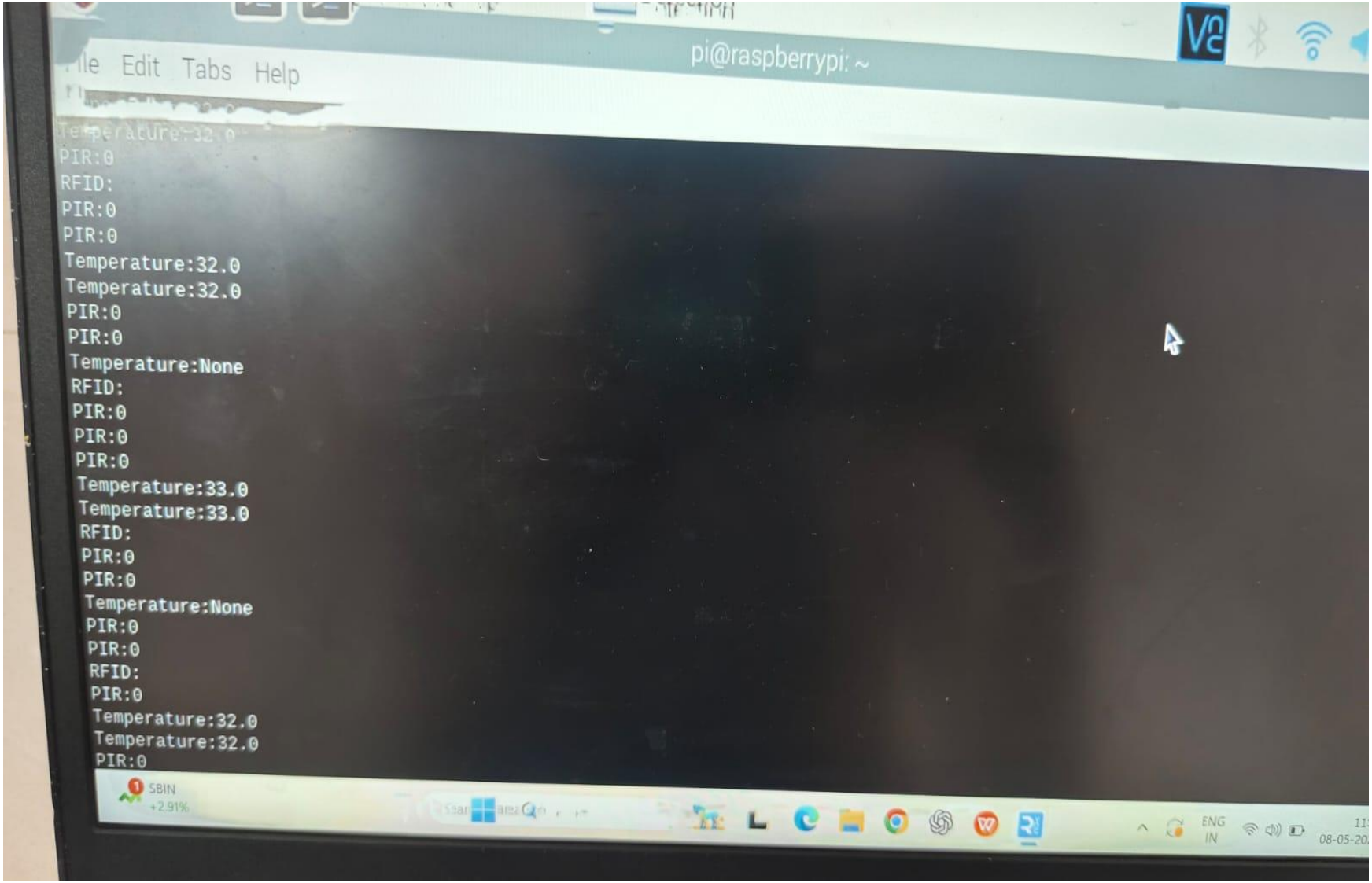
APPENDIX I

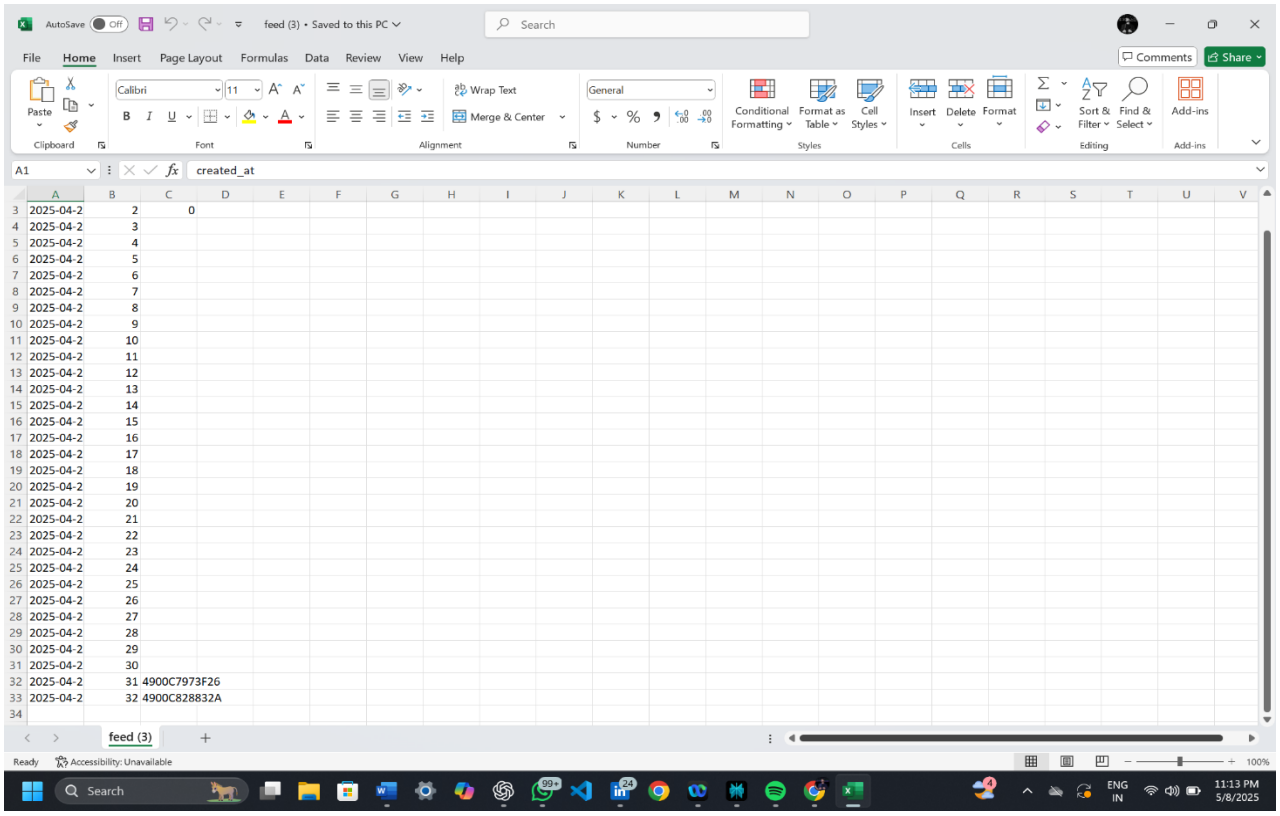
```
# Required Libraries
import torch
import cv2
import time
import os

# Load YOLOv5 model (Pretrained or Custom trained for smartphone detection)
model = torch.hub.load('ultralytics/yolov5', 'custom', path='smartphone_model.pt')  # Replace with your trained model

# Load Webcam or Video File
cap = cv2.VideoCapture(0)  # Use 0 for webcam or provide path to video file

# Define Buzzer Simulation Function
def buzzer_alert():
    print("🔔 Alert: Smartphone Detected in Classroom!")
    # You can connect GPIO for real buzzer using Raspberry Pi or Arduino

# Run Detection in Real-Time
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Perform detection
    results = model(frame)
```

```
    # Parse Results
    detections = results.pandas().xyxy[0]  # Bounding boxes with labels and confidence

    # Filter for 'smartphone' label (Ensure your model uses correct label names)
    for index, row in detections.iterrows():
        if row['name'] == 'smartphone' and row['confidence'] > 0.5:
            # Draw Bounding Box
            x1, y1, x2, y2 = int(row['xmin']), int(row['ymin']), int(row['xmax']),
int(row['ymax'])
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
            cv2.putText(frame, 'Smartphone', (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)

            # Trigger Buzzer Alert
            buzzer_alert()

    # Display the Frame
    cv2.imshow('Smartphone Detection', frame)

    # Exit on 'q' Key
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release Resources
cap.release()
cv2.destroyAllWindows()
```

APPENDIX II

OUTPUT SNIPPETS


File Edit Tabs Help
pi@raspberrypi: ~
PIR:0
RFID:
PIR:0
PIR:0
Temperature:32.0
Temperature:32.0
PIR:0
PIR:0
Temperature:None
RFID:
PIR:0
PIR:0
PIR:0
Temperature:33.0
Temperature:33.0
RFID:
PIR:0
PIR:0
Temperature:None
PIR:0
PIR:0
RFID:
PIR:0
Temperature:32.0
Temperature:32.0
PIR:0
SBIN
+2.91%
ENG
IN



	A	B	C
3	2025-04-2	2	0
4	2025-04-2	3	
5	2025-04-2	4	
6	2025-04-2	5	
7	2025-04-2	6	
8	2025-04-2	7	
9	2025-04-2	8	
10	2025-04-2	9	
11	2025-04-2	10	
12	2025-04-2	11	
13	2025-04-2	12	
14	2025-04-2	13	
15	2025-04-2	14	
16	2025-04-2	15	
17	2025-04-2	16	
18	2025-04-2	17	
19	2025-04-2	18	
20	2025-04-2	19	
21	2025-04-2	20	
22	2025-04-2	21	
23	2025-04-2	22	
24	2025-04-2	23	
25	2025-04-2	24	
26	2025-04-2	25	
27	2025-04-2	26	
28	2025-04-2	27	
29	2025-04-2	28	
30	2025-04-2	29	
31	2025-04-2	30	
32	2025-04-2	31	4900C7973F26
33	2025-04-2	32	4900C828832A
34			

REFERENCES

1. Kaur, H., & Singh, G. (2021). *IoT and AI-Based Attendance and Mobile Usage Monitoring System in Classrooms*. International Journal of Recent Technology and Engineering, 9(6), 210–215.
2. Srivastava, N., & Suri, P. K. (2021). *AI and Deep Learning Techniques for Smart Surveillance: A Review*. ACM Computing Surveys, 54(3), 1–35.
3. Rasool, A., & Javed, H. (2021). *Smart Surveillance System Using AI for Monitoring Classroom Activity*. International Journal of Advanced Computer Science and Applications, 12(4), 198–205.
4. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2021). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv preprint arXiv:2004.10934.
5. Jocher, G., et al. (2021). *YOLOv5 by Ultralytics*. GitHub Repository. Retrieved from
6. Zhang, Y., & Zhao, J. (2022). *Intelligent Classroom Monitoring Using Object Detection and Pose Estimation*. International Journal of Computer Applications, 184(18), 22–28.
7. Twilio Inc. (2023). *Twilio Messaging API Documentation*. Retrieved from [https://www.twilio.com/docs/sms/send-messages](https://www.twilio.com/docs/sms/send-m
8. Li, X., & Kumar, P. (2021). *Deep Learning-based Surveillance System for Unauthorized Smartphone Usage Detection*. International Journal of Artificial Intelligence Research, 5(2), 95–103.
9. Gupta, R., & Mehta, S. (2021). *YOLO-Based Real-Time Object Detection for Monitoring Mobile Phone Use in Smart Classrooms*. International Conference

on Computer Vision and Smart Technologies (ICCVST), IEEE, pp. 122–127.

10. Sharma, A., & Mishra, D. (2022). *AI-powered Edge Devices for Real-Time Classroom Monitoring*. Journal of Smart Educational Systems, 10(1), 45–53.

11. Prasad, M. N., & Reddy, B. S. (2023). *Integration of IoT and AI in Academic Surveillance Systems*. Journal of Internet of Things and Applications, 6(1), 88–96.

12. Patel, H., & Agarwal, N. (2023). *A YOLOv5-based Lightweight Model for Smartphone Detection in Restricted Zones*. Proceedings of the 2023 International Conference on Embedded Systems and AI (ICESAI), pp. 90–97.

13. NVIDIA Developer. (2023). *Deploying YOLOv5 on Jetson Nano for Real-Time Inference*. Retrieved from

14. AWS IoT Core. (2023). *AWS IoT Core Documentation: Connecting Edge Devices to the Cloud*. Retrieved from

15. Firebase. (2023). *Firebase Realtime Database for IoT Alerts and Notifications*.

16. Reddy, A., & Joshi, V. (2022). *AI-Based Classroom Monitoring Using Object Detection Techniques*. International Journal of Advanced Trends in Computer Science and Engineering, 11(5), 312–319.

17. Wang, T., & Liu, X. (2021). *Real-Time Smartphone Detection Using YOLOv4-Tiny in School Environments*. Journal of Real-Time Image Processing, 18(4), 401–412.

18. Khan, F., & Sharma, L. (2022). *Smart Education Surveillance Using Deep Learning and IoT*. Procedia Computer Science, 199, 1002–1010.

19. Bose, S., & Jain, R. (2023). *AI and IoT Fusion for Smart Campus Monitoring*. International Journal of Computer Applications in Technology, 73(2), 110–117.

20. Nguyen, T. H., & Vo, Q. M. (2021). *Efficient Object Detection Using YOLO for Edge-Based Monitoring Systems*. Sensors, 21(12), 3987.