**CHAPTER 1**

**INTRODUCTION**

The increasing incidence of vehicle theft poses a significant challenge to vehicle owners and law enforcement agencies worldwide. In recent years, the sophistication and ingenuity of thieves have escalated, rendering traditional security measures, such as locks and alarms, increasingly inadequate. Modern thieves employ advanced techniques and tools, such as electronic key cloning, signal jamming, and relay attacks, to bypass conventional security systems with alarming ease. These methods can often go undetected until it is too late, leading to substantial financial losses and emotional distress for vehicle owners.

The traditional mechanical locks and alarms, while still in use, offer only a basic level of protection. Mechanical locks can be picked or broken, and alarms can be disabled or ignored. Additionally, the widespread use of electronic key fobs has introduced new vulnerabilities, such as relay attacks, where thieves use signal boosters to intercept and amplify the key fob signal, enabling them to unlock and start the vehicle remotely. This method requires no physical break-in, making it difficult to detect and prevent.

Given these sophisticated threats, there is a pressing need for more advanced and comprehensive security solutions. Our proposed anti-theft device aims to address these challenges by integrating a combination of cutting-edge technologies to provide a robust three-tiered security system. The primary objective of this device is to substantially reduce the risk of vehicle theft by implementing a multi-layered approach to security. Each layer is meticulously designed to address different aspects of vehicular safety, ensuring comprehensive protection against unauthorized access and theft.

At the heart of the device is the RFID system, which serves as the first level of defence by verifying the presence of an authorized tag before enabling the ignition. This measure ensures that only individuals with the correct RFID tag can start the vehicle, effectively preventing unauthorized access. The second level of security utilizes an accelerometer to detect any unauthorized movement of the vehicle, triggering alerts to prevent potential theft. This feature is particularly effective in detecting attempts to tow or physically move the vehicle without authorization.

The third level of security integrates GPS and GSM technologies, allowing for real-time tracking of the vehicle's location and immediate communication with the owner in case of any security breach. This combination not only facilitates the quick recovery of stolen vehicles but also serves as a powerful deterrent to thieves who are aware of the tracking capabilities. The real-time alerts enable the owner to take swift action, such as contacting law enforcement, to recover the vehicle before it is moved to a different location or dismantled.

By deploying this innovative anti-theft device on a two-wheeler scooter (Activa) and observing its performance over several weeks, we have confirmed its efficiency and practicality. The positive feedback received underscores its potential as a valuable tool for vehicle owners seeking to protect their assets. This device's ability to provide multiple levels of security, including authentication, movement detection, and real-time tracking, offers a comprehensive solution to address the evolving challenges of vehicle theft.

The consequences of vehicle theft extend beyond financial losses, impacting the psychological well-being and daily lives of vehicle owners. The aftermath of such incidents often involves lengthy and stressful interactions with insurance companies, potential hikes in insurance premiums, and the inconvenience of being without a primary mode of transportation. Moreover, the personal value attached to vehicles, which often hold sentimental significance or serve as critical tools for livelihood, amplifies the distress experienced by owners. Hence, a sophisticated anti-theft solution that can effectively mitigate these risks is not just a luxury but a necessity in today's world.

In developing our anti-theft device, we prioritized ease of use and accessibility. The device is designed to be user-friendly, ensuring that vehicle owners can easily manage and operate the security features without requiring extensive technical knowledge. The installation process is straightforward, and the device can be integrated into various types of vehicles, from scooters to cars and even commercial vehicles. This versatility makes it a practical solution for a broad audience, addressing diverse security needs across different vehicle categories.

In essence, the proposed anti-theft device not only addresses current vehicle theft threats with a multi-layered security approach but also lays the groundwork for future advancements in this critical area. Its practical application and positive reception highlight its potential to significantly reduce vehicle theft incidents, providing peace of mind to vehicle owners. By continuing to innovate and collaborate, we can ensure that our vehicles remain secure, safeguarding our assets and contributing to overall public safety.

**Introduction to Embedded System**

Embedded systems play a pivotal role in the modern technological landscape, serving as the backbone of numerous applications ranging from consumer electronics to industrial automation. An embedded system is a specialized computing system designed to perform dedicated functions within a larger system. Unlike general-purpose computers, embedded systems are tailored for specific tasks, offering optimized performance, reliability, and efficiency. They integrate hardware and software components to execute predefined functions, often operating under real-time constraints to ensure timely and predictable responses.

At the core of an embedded system is a microcontroller or microprocessor, which acts as the central processing unit (CPU). These processors are designed to manage a variety of inputs and outputs, control peripheral devices, and execute embedded software programs. The choice of processor depends on the complexity and requirements of the application, with options ranging from simple 8-bit microcontrollers to advanced 32-bit microprocessors capable of handling sophisticated tasks. Alongside the CPU, embedded systems incorporate memory components, input/output interfaces, sensors, and actuators, all working together to fulfil the system’s specific functions.

The versatility of embedded systems is evident in their widespread application across various domains. In the automotive industry, embedded systems are integral to the operation of anti-lock braking systems (ABS), airbag controllers, and infotainment systems. In consumer electronics, they power devices such as smartphones, smart TVs, and home automation systems. Industrial applications include robotics, process control systems, and smart grid technologies. Additionally, embedded systems are crucial in healthcare, enabling the functionality of medical devices like pacemakers, diagnostic equipment, and patient monitoring systems.

Embedded systems are characterized by several key attributes that distinguish them from general-purpose computing systems. These include real-time operation, where the system must process data and respond within strict time constraints; low power consumption, crucial for battery-operated devices; and high reliability, essential for mission-critical applications. Furthermore, embedded systems often operate in constrained environments, necessitating efficient resource management and robust security measures to protect against potential vulnerabilities.

The development of embedded systems involves a multidisciplinary approach, combining expertise in electronics, computer science, and software engineering. Engineers must design both the hardware architecture and the software algorithms to ensure seamless integration and optimal performance. This includes writing firmware, which is the low-level software that directly interacts with the hardware, as well as developing higher-level application code. Tools such as integrated development environments (IDEs), simulators, and debuggers are essential in the development process, providing the means to design, test, and refine embedded systems efficiently.

**Definition of Embedded system**

An embedded system is a specialized computer system designed to perform a specific task within a larger device or system. Unlike a general-purpose computer designed for various tasks, an embedded system is dedicated to a particular function and typically doesn't have a user interface like a monitor, keyboard, or mouse.

**Features of embedded system**

Embedded systems are characterized by several key features that differentiate them from general-purpose computing systems. These features are designed to meet the specific requirements of the tasks they are intended to perform, ensuring optimal performance, reliability, and efficiency. Here are some of the prominent features of embedded systems:

1. **Real-Time Operation:** Embedded systems often need to operate under real-time constraints, meaning they must process inputs and produce outputs within a strict timeframe. This is critical for applications such as automotive control systems, medical devices, and industrial automation, where timely responses are essential for safety and functionality.
2. **Dedicated Functionality:** Unlike general-purpose computers, embedded systems are designed for specific tasks. This focused design allows for optimization of both hardware and software to achieve high efficiency and performance for the intended application.
3. **Low Power Consumption:** Many embedded systems, particularly those in portable or battery-operated devices, must minimize power consumption to extend battery life. Energy-efficient hardware design and power management strategies are integral to achieving this goal.
4. **Compact Size:** Embedded systems are typically designed to be compact and lightweight, making them suitable for integration into a wide variety of products and environments. This is particularly important in consumer electronics, medical devices, and wearable technology.
5. **Reliability and Stability**: High reliability and stability are critical features of embedded systems, especially in applications where failure could lead to significant consequences, such as in aerospace, medical, and industrial systems. Embedded systems are often designed with robust error handling and redundancy to ensure continuous operation.
6. **Specific Hardware and Software Requirements**: Embedded systems are designed with specific hardware components, such as microcontrollers, sensors, and actuators, tailored to the needs of the application. The software, typically referred to as firmware, is developed to directly control and interact with this hardware, ensuring seamless operation.
7. **Efficient Resource Utilization**: Embedded systems often operate with limited resources, such as memory, processing power, and storage. Efficient use of these resources is critical, requiring careful design and optimization of both hardware and software.
8. **Integration with Other Systems:** Embedded systems frequently need to interface and communicate with other systems and devices. This can involve various communication protocols, such as I2C, SPI, UART, CAN, Ethernet, and wireless standards like Wi-Fi, Bluetooth, and Zigbee.
9. **Security:** Security is a significant concern for embedded systems, particularly those involved in sensitive or critical applications. This includes ensuring data integrity, protecting against unauthorized access, and safeguarding the system from cyber-attacks.
10. **Scalability and Flexibility:** While embedded systems are often designed for specific tasks, there is sometimes a need for scalability and flexibility to accommodate future upgrades or changes in functionality. Modular design approaches and the use of configurable software and hardware components help address these needs.
11. **Cost-Effectiveness:** Cost is a crucial factor in the design and deployment of embedded systems. Engineers strive to achieve a balance between performance and cost, selecting components and designing systems that meet the necessary specifications while remaining economically viable.
12. **Long Lifespan and Maintenance:** Many embedded systems are designed for long-term use, often in environments where maintenance or replacement is challenging. As such, they need to be durable and capable of operating reliably over extended periods, sometimes requiring the ability to update firmware remotely (over-the-air updates).

**Applications of Embedded system**

Embedded systems are integral to a wide array of applications across various industries, demonstrating their versatility and critical importance in modern technology. Here are some notable applications:

**Consumer Electronics:**

* **Mobile devices:** Smartphones, tablets, wearables (smartwatches, fitness trackers) all rely on embedded systems for core functionalities like touchscreens, cameras, sensors, and efficient power management.
* **Home appliances:** Smart TVs, refrigerators, washing machines, and thermostats often have embedded systems that control features, connect to Wi-Fi, and optimize performance.
* **Gaming consoles:** Embedded systems power the processing, graphics, and user input functionalities within gaming consoles.
* **Audio/Video devices:** Digital cameras, music players, and video editing tools utilize embedded systems for image/sound processing and control operations.

**Industrial Automation:**

* **Industrial robots:** Embedded systems are the brains of robots used in manufacturing, assembly lines, and material handling. They control movements, interact with sensors, and ensure precise operation.
* **Factory machinery:** Programmable logic controllers (PLCs) are essentially powerful embedded systems that automate and control various industrial machines and processes.
* **Building automation systems:** Embedded systems manage heating, ventilation, air conditioning (HVAC) systems, lighting control, and security systems in buildings.

**Automotive Systems:**

* **Electronics control units (ECUs):** These embedded systems manage various aspects of an engine's operation, including fuel injection, ignition timing, and emission control.
* **Safety systems:** Anti-lock braking systems (ABS), airbags, and traction control systems rely on embedded systems for real-time monitoring and critical actions during emergencies.
* **Driver assistance systems:** Features like lane departure warning, blind-spot monitoring, and adaptive cruise control utilize embedded systems with sensors and cameras for improved safety and driver assistance.

**Internet of Things (IoT):**

* **Smart home devices:** Smart thermostats, connected doorbells, and smart plugs use embedded systems to collect sensor data, communicate over networks, and enable remote control and automation within homes.
* **Wearable technology:** Fitness trackers and smartwatches often have embedded systems to monitor heart rate, activity levels, and sleep patterns.
* **Industrial IoT (IoT):** Sensors and devices in industrial settings use embedded systems to collect data on machine performance, environmental conditions, and enable remote monitoring and optimization of operations.

**Medical Devices:**

* **Pacemakers:** These embedded systems regulate heart rhythm by sending electrical signals to the heart muscle.
* **Insulin pumps:** These devices use embedded systems to monitor blood sugar levels and deliver precise insulin doses as needed.
* **Diagnostic equipment:** Embedded systems are used in medical imaging devices (MRI, CT scans) for data acquisition, processing, and image generation.

**Other Applications:**

* **Transportation systems:** Traffic light controllers, ticketing systems, and navigation systems in airplanes and trains all rely on embedded systems for operation and control.
* **Communication systems:** Embedded systems are within routers, switches, and base stations that form the backbone of communication networks.
* **Military and aerospace:** Embedded systems play a crucial role in weapon systems, navigation systems, and flight control systems in aircrafts and missiles.

**Types of Embedded system**

Embedded systems can be classified into various categories based on different characteristics.

Some common types of embedded systems:

**By Function:**

* **Stand-alone embedded systems:** These systems operate independently without relying on other devices. They typically have their own user interface (like buttons and LEDs) for user interaction. An example is a simple calculator or a microwave oven.
* **Networked embedded systems:** These systems rely on a network connection (wired or wireless) to communicate with other devices or a central server. They often don't have a user interface and exchange data with other systems to perform their tasks. Examples include smart thermostats, network routers, and industrial control systems.

**By Performance:**

* **Small-scale embedded systems:** These systems have minimal processing power, memory, and storage needs. They are often used in simple devices like remote controls, sensors, and wearables.
* **Medium-scale embedded systems:** These systems have more processing power and memory than small-scale systems and can handle more complex tasks. They are commonly used in printers, routers, and industrial controllers.
* **Sophisticated embedded systems:** These systems are high-performance embedded systems with significant processing power, memory, and storage capabilities. They are used in applications requiring complex calculations, real-time processing, and advanced features. Examples include medical devices, avionics (aviation electronics), and self-driving cars.

**By Real-Time Constraints:**

* **Soft real-time systems:** These systems strive to respond to events within a reasonable time frame, but missing deadlines occasionally might not be critical. Multimedia players, traffic signal controllers, and voice processing systems are some examples.
* **Hard real-time systems:** These systems have strict timing requirements. Missing deadlines in these systems can have severe consequences. Real-time control systems in industrial automation, anti-lock braking systems in cars, and airbag deployment systems are examples of hard real-time systems.

**Other Categories:**

* **Mobile embedded systems:** These systems are embedded within portable devices like smartphones, tablets, and drones. They are often battery-powered and require efficient use of resources.

**CHAPTER 2**

**METHODOLOGY**

The following points represents the path taken to reach the project’s goal.

**Block Diagram of the components involved:**

GSM

ENGINE

GPS

RELAY

POWER SUPPLY

RFID

ACCELEROMETER

ARDUINO UNO

(MCU)

Fig1: Functional block diagram of the device.

**Hardware**

* **Arduino UNO**

Arduino is a popular choice for beginners due to its ease of use and large community, but there are many other microprocessors available for various purposes. Here are some alternatives to Arduino, categorized based on their strengths:

**Some similar microprocessors:**

1. **Raspberry Pi Pico:** This microcontroller offers similar features to Arduino Uno but with a more powerful processor and ability to run a basic operating system. It's a good option for those wanting to transition from basic Arduino projects to explore microcontrollers with more processing power. It is more expensive.
2. **ESP32 Dev Kit C:** This development board features an ESP32 microcontroller that includes Wi-Fi and Bluetooth connectivity built-in. It's a great choice for projects involving internet connectivity or wireless communication between devices. The Arduino IDE can be used to program the ESP32 with some additional libraries but It has less ports.

* **Power Supply**

It is an electrical device that converts electric current from a source to the correct voltage, current, and frequency to power an electrical load (device). It's essentially an intermediary that ensures your electronic devices receive the appropriate electrical power to function correctly.

* **RFID MFRC522**

The MFRC522 is a popular choice for RFID applications due to its affordability and ease of use, but there are several other RFID reader chips available depending on your project needs. It is more reliable.

Here are some alternatives to the MFRC522:

**PN532:** This chip is a more versatile option compared to the MFRC522. It supports not only Mifare Classic (like MFRC522) but also other popular RFID standards like Mifare Ultralight, FeliCa, and ISO/IEC 14443A/B. Additionally, it has built-in Near Field Communication (NFC) capabilities, making it suitable for projects involving contactless payments or data exchange.

* **Accelerometer ADXL345**

The ADXL345 is a popular choice for accelerometers due to its affordability, ease of use, and good performance, but there are many other accelerometers available depending on your project's needs.

Here are some alternatives to the ADXL345:

**LIS3DH:** This is a common alternative to the ADXL345 with similar features like 3-axis acceleration measurement, selectable g-ranges (typically ±2g or ±4g), and SPI or I2C communication interfaces. It's a good option for projects requiring basic motion detection or tilt sensing.

**MMA7660:** This is another 3-axis accelerometer with selectable g-ranges and SPI or I2C interfaces. It offers a lower power consumption compared to the ADXL345, making it suitable for battery-powered applications.

* **GSM & GPS module Sim808**

The SIM808 is a popular GSM/GPRS module with built-in GPS functionality, but there are several alternatives depending on your project's needs. It is widely available and cost effective.

**SIM7000 Series (SIMCom):** This series offers a wide range of cellular modules supporting various protocols like GSM, GPRS, EDGE, UMTS, and LTE (4G). They are a good choice for projects requiring different cellular network connectivity options.

**ESP32 Modules with Cellular Connectivity:** Some ESP32 development boards come integrated with cellular connectivity options like GSM or LTE. This combines a powerful microcontroller (ESP32) with cellular capabilities in a single package.

* **5V Relay**

It is widely available, compatible with Arduino and cost effective.

* **Jumper wires**

It is used to connect various components.

**Software**

* **Arduino IDE**

Arduino IDE (Integrated Development Environment) is a free, open-source software application that makes writing code and uploading it to Arduino boards incredibly easy. It's a fantastic platform for beginners and experienced programmers alike, especially for those interested in electronics, robotics, and physical computing.

* **Embedded C++ language**

Arduino is an open-source electronics platform that makes it easier to write code and upload it to physical electronic boards. While Arduino uses a simplified version of C++, it's not strictly embedded C++.

**CHAPTER 3**

**HARDWARE DESCRIPTION**

**3.1 Arduino UNO**

Arduino UNO is a microcontroller board based on the **ATmega328P**. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Arduino is an open-source electronics platform consisting of easy-to-use hardware and software. It's intended for anyone wanting to learn electronics, coding, or tinkering with interactive projects.

Here are some key features of Arduino:

* **Hardware (Arduino Boards):** Arduino boards are single-board microcontrollers, meaning they contain all the electronic components needed to operate on a printed circuit board (PCB). These boards come in various shapes and sizes, with varying capabilities depending on the project's needs. A popular board for beginners is the Uno, which has 14 digital input/output pins (of which 6 provide pulse-width modulation (PWM) output), 6 analog input pins, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

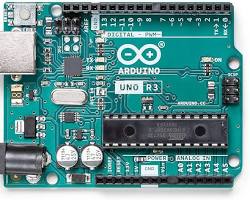


Fig2: Arduino UNO board.

* **Software (Arduino IDE):** Arduino IDE (Integrated Development Environment) is a free, open-source software application that makes it easy to write code and upload it to the Arduino board. It simplifies the process of working with microcontrollers, eliminating the need for users to write complex low-level code or interface with a separate programmer.



Fig3: Arduino IDE.

* **Easy-to-learn programming language:** Arduino uses a simplified version of C++, making it easier to learn for beginners with no prior coding experience.
* **Open-source and large community:** Arduino is open-source, which means the hardware reference designs and software source code are freely available for anyone to use, modify, or distribute. This has fostered a large and active community of Arduino users who share projects, code libraries, and tutorials.
* **Versatility:** Arduino boards are incredibly versatile and can be used for various projects, including robotics, home automation, scientific data collection, wearable electronics, and much more. With the right components, you can build nearly anything you can imagine.

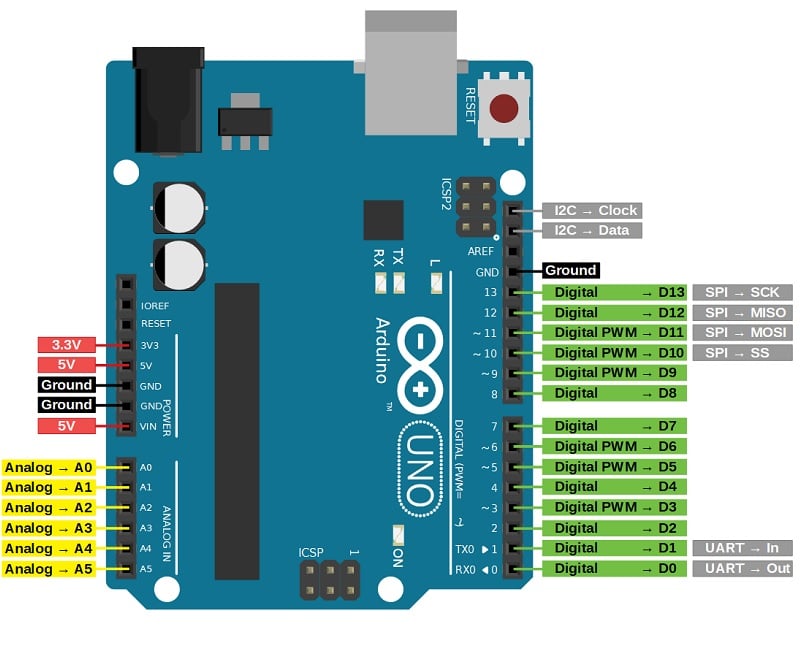


Fig4: Arduino UNO board showing various pins.

Arduino Uno pins are directly connected to the microcontroller (ATmega328P in most Uno versions) and categorized into three types: digital pins, analog pins, and special function pins. While the Arduino software refers to them by pin numbers (0 to 13 for digital, A0 to A5 for analog), these pins are also connected to specific ports on the microcontroller chip. Understanding these ports is useful for advanced users who want more control over the hardware.

Here's a breakdown of the Arduino Uno ports and their corresponding pins:

**1. Digital Pins (PORTD and PORTB):**

* Arduino Uno has 14 digital pins (0 to 13). These pins can be used for both input and output operations.
* Pins 0 and 1 (PORTD) are also used for serial communication.
* Pins 3, 5, 6, 9, 10, and 11 (PORTD) have built-in PWM (Pulse Width Modulation) capabilities for analog-like control of outputs like LEDs or motors.
* Pins 8 to 13 (PORTB) are for general digital input/output.

**2. Analog Pins (PORTC):**

* Arduino Uno has 6 analog pins (A0 to A5). These pins can read analog voltage signals from sensors (0 to 5 volts).
* Internally, these pins are connected to an analog-to-digital converter (ADC) within the microcontroller.

**3. Special Function Pins:**

* These pins have specific purposes beyond general input/output.
* AREF (Analog Reference): Used to provide an external reference voltage for the analog-to-digital converter.
* GND (Ground): Several ground pins are available to connect various components to the common ground level of the circuit.
* VCC (Voltage Source): Several VCC pins provide 5V power to components.
* 3V3 (3.3V Power): Provides 3.3V power for certain components.
* RESET: Resets the microcontroller.
* ICSP (In-Circuit Serial Programming): Allows uploading new code to the Arduino board without removing it from the circuit.

To set up the Arduino IDE on our computer and prepare to connect our Arduino Uno via cable:

**1. Download and Install Arduino IDE:**

* Visit the official Arduino website <https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE>.
* Click on the "**Download the Arduino IDE**" button.
* Choose the appropriate installer for our operating system (Windows, macOS, Linux).
* Download and run the installer, following the on-screen instructions.

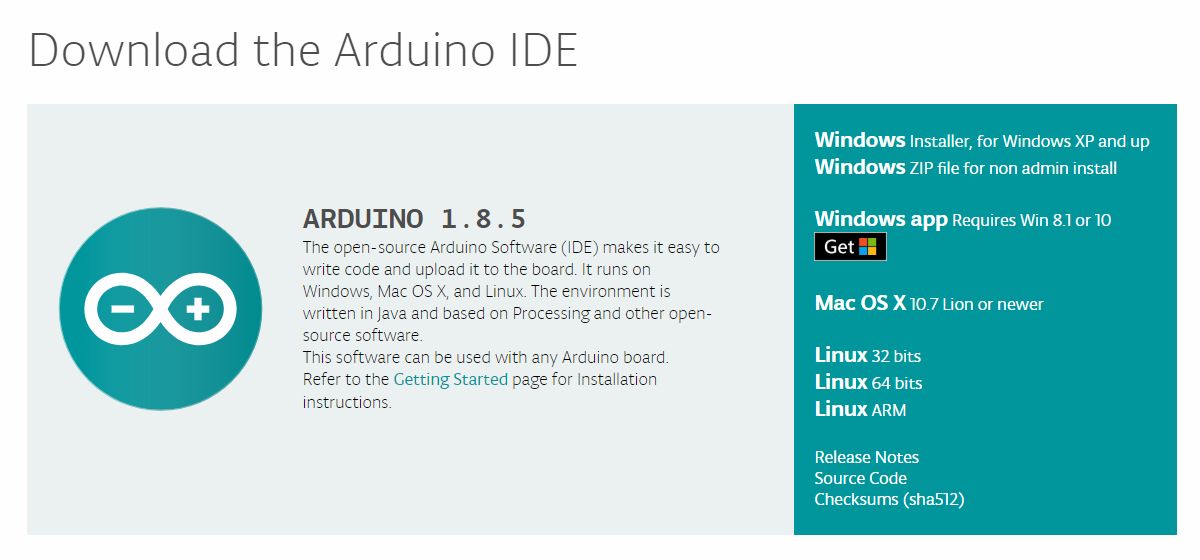


Fig5: Arduino IDE download.

**2. Connect Your Arduino Uno (Optional - Driver Installation):**

* Before installing drivers, connect your Arduino Uno to your computer using a USB cable.



Fig6: Type A to Type B connecting cable.

* In some cases, our computer might automatically detect and install the necessary drivers.
* If the drivers aren't installed automatically, follow these steps (specific instructions may vary slightly depending on your OS):
  + Open the **Device Manager** (search for it in your start menu/applications).

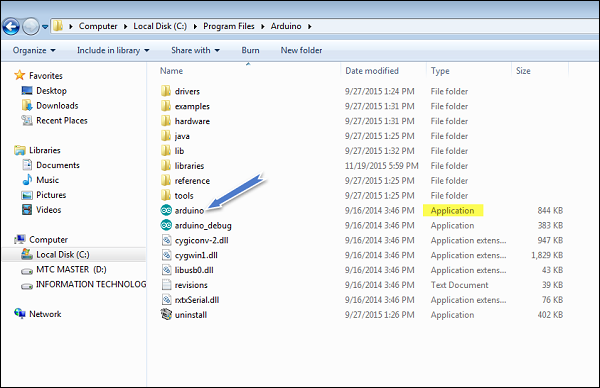


Fig7: Launching Arduino.

* + Look for the **Ports (COM & LPT)** section. You might see "Arduino Uno (COMx)" listed here (x represents a number).
  + If you see an "Unknown Device" instead, right-click on it and select "**Update Driver Software**".
  + Choose "**Browse my computer for driver software**".
  + Navigate to the Arduino IDE installation directory (usually under "Documents\Arduino\drivers" on Windows).
  + Select the driver file (typically named "arduino.inf") and click "**Next**".
  + Windows will install the drivers.

**3. Verify Board Selection in Arduino IDE:**

* Open the Arduino IDE software.
* Go to **Tools > Board** menu.
* Select the appropriate board you have connected. In this case, choose "**Arduino Uno**".

**4. Verify Serial Port Selection (Optional):**

* In some cases, the Arduino IDE might automatically detect the serial port your Arduino Uno is connected to.
* You can verify this by going to **Tools > Port**.
* If you see "Arduino Uno (COMx)" listed (where x is a number), that's the correct port.
* If you don't see the port, try unplugging and replugging the Arduino Uno and refreshing the list in the Arduino IDE.

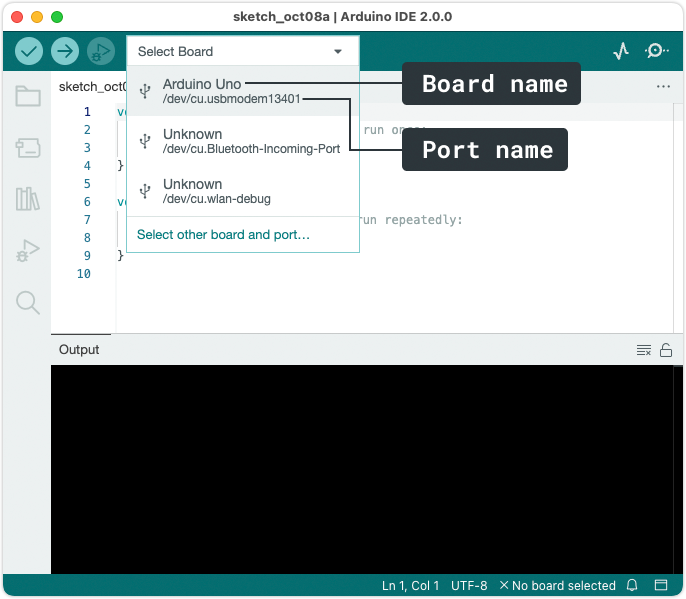


Fig8: Choosing board and port name.

**5. Uploading a Simple Test Sketch (Optional):**

* This step is optional but helps verify your setup.
* Open the "Blink" example sketch included with the Arduino IDE. Go to **File > Examples > Basic > Blink**.
* The sketch blinks an LED connected to pin 13.
* Make sure your Arduino Uno is connected and the correct board and port are selected (as mentioned in steps 3 and 4).
* Click the **Upload** button (looks like an arrow pointing to the board) in the Arduino IDE toolbar.
* If everything is set up correctly, the sketch will upload to your Arduino Uno, and the LED should start blinking.

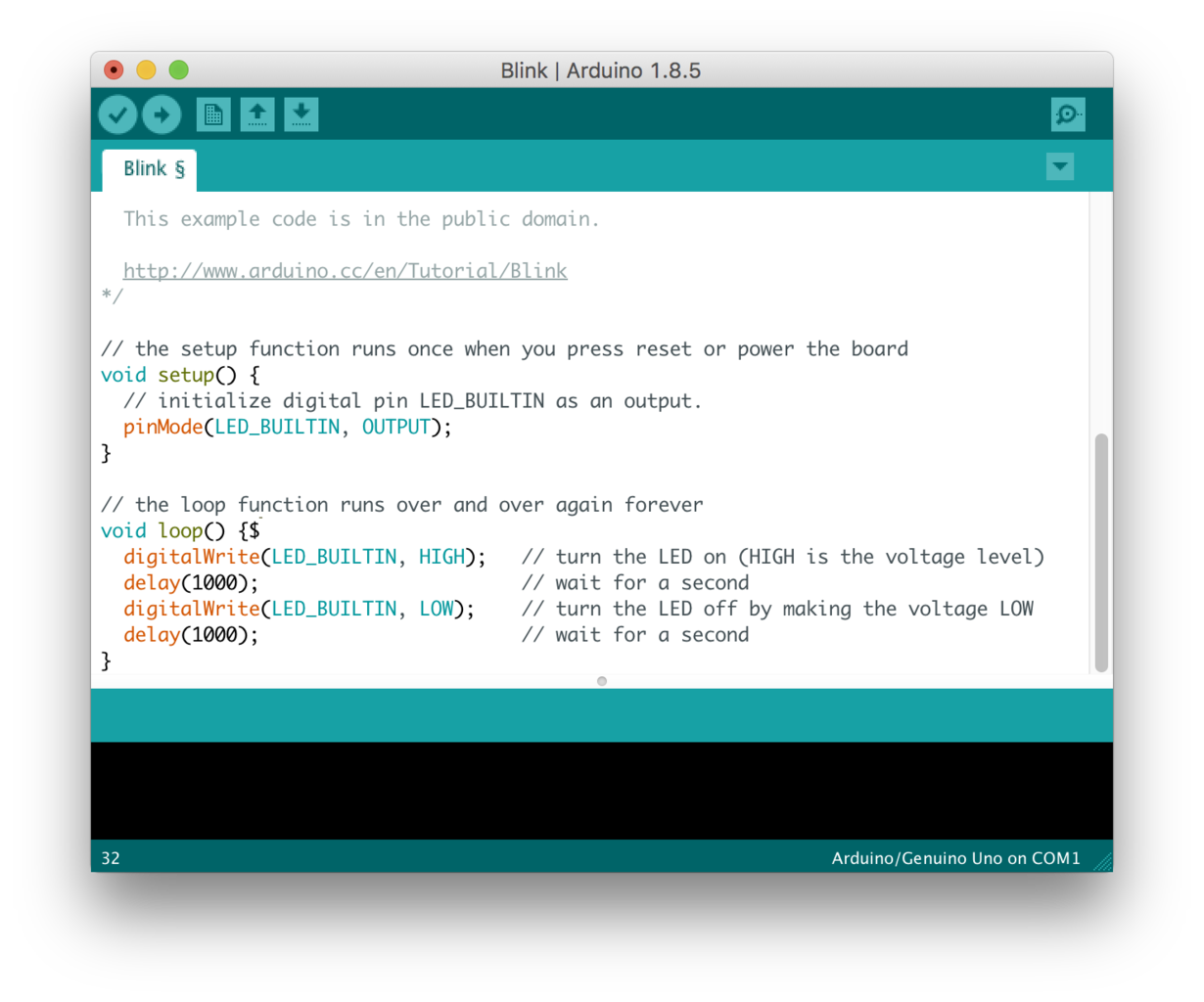


Fig9: Basic blink code to check Arduino is working.

**3.2 RFID MFRC522**

The MFRC522 is a highly integrated reader/writer IC for contactless communication at 13.56 MHz. The MFRC522 reader supports ISO/IEC 14443 A/MIFARE and NTAG.

The MFRC522’s internal transmitter is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443 A/MIFARE cards and transponders without additional active circuitry. The receiver module provides a robust and efficient implementation for demodulating and decoding signals from ISO/IEC 14443 A/MIFARE compatible cards and transponders. The digital module manages the complete ISO/IEC 14443 A framing and error detection (parity and CRC) functionality.

The MFRC522 supports MF1xxS20, MF1xxS70 and MF1xxS50 products. The MFRC522 supports contactless communication and uses MIFARE higher transfer speeds up to 848 kBd in both directions.

The following host interfaces are provided:

* Serial Peripheral Interface (SPI)
* Serial UART (similar to RS232 with voltage levels dependant on pin voltage supply)
* I 2C-bus interface



Fig10: MFRC522 with RFID tag.

**Features and benefits**

* Highly integrated analog circuitry to demodulate and decode responses
* Buffered output drivers for connecting an antenna with the minimum number of external components
* Supports ISO/IEC 14443 A/MIFARE and NTAG
* Typical operating distance in Read/Write mode up to 50 mm depending on the antenna size and tuning
* Supports MF1xxS20, MF1xxS70 and MF1xxS50 encryption in Read/Write mode
* Supports ISO/IEC 14443 A higher transfer speed communication up to 848kBd
* Supports MFIN/MFOUT
* Additional internal power supply to the smart card IC connected via MFIN/MFOUT
* Supported host interfaces
* SPI up to 10 Mbit/s
* I 2C-bus interface up to 400 kBd in Fast mode, up to 3400 kBd in High-speed mode
* RS232 Serial UART up to 1228.8 kBd, with voltage levels dependant on pin voltage supply
* FIFO buffer handles 64 byte send and receive
* Flexible interrupt modes
* Hard reset with low power function
* Power-down by software mode
* Programmable timer
* Internal oscillator for connection to 27.12 MHz quartz crystal
* 2.5 V to 3.3 V power supply
* CRC coprocessor
* Programmable I/O pins
* Internal self-test

**Block Diagram**

The analog interface handles the modulation and demodulation of the analog signals.

The contactless UART manages the protocol requirements for the communication protocols in cooperation with the host. The FIFO buffer ensures fast and convenient data transfer to and from the host and the contactless UART and vice versa.

Various host interfaces are implemented to meet different customer requirements.

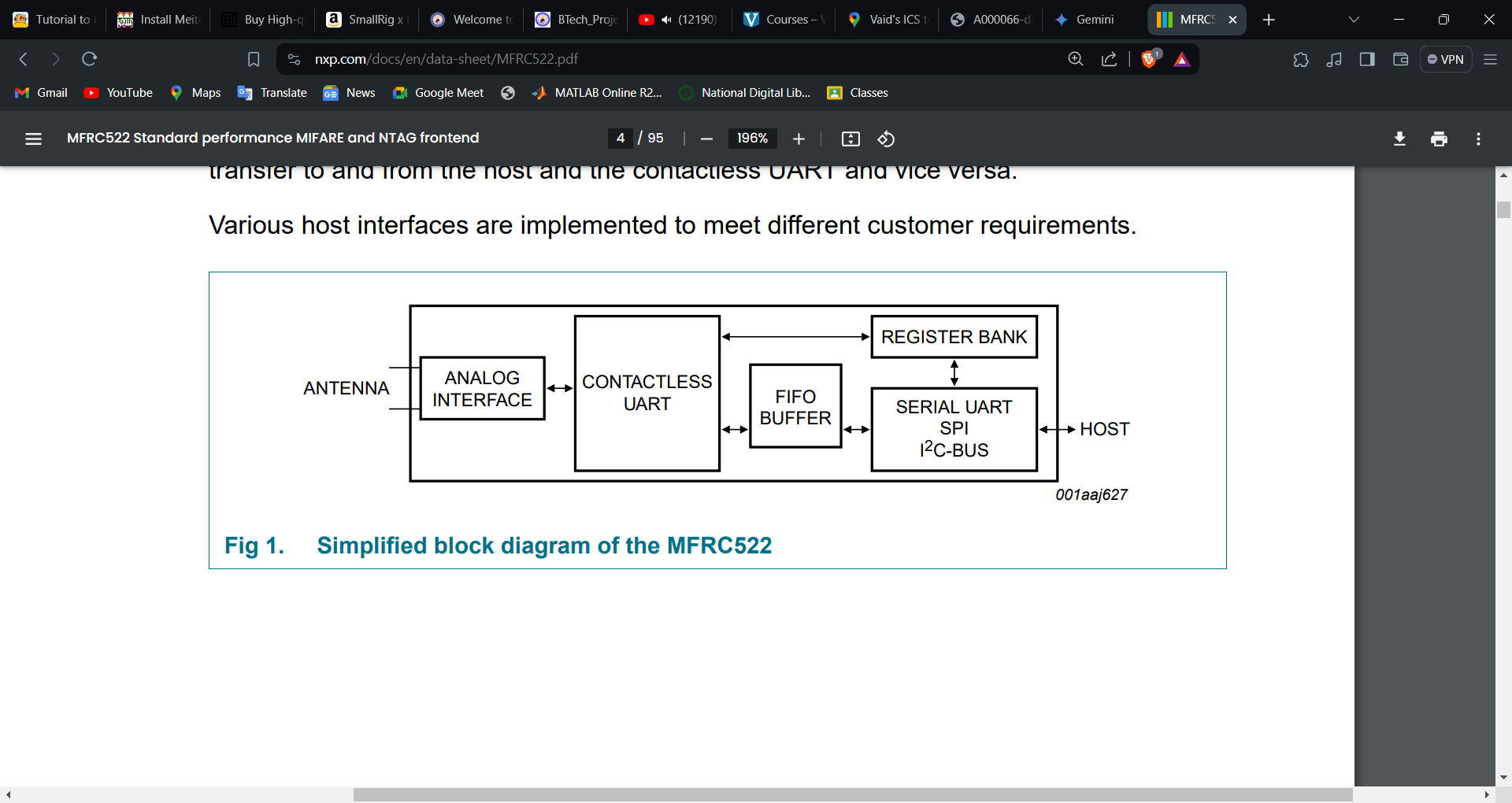


Fig11: Simplified block diagram of MFRC522.

**Pinning information of MFRC522**

The MFRC522 RFID reader module itself doesn't have traditional ports like USB or HDMI that you might find on a computer. It communicates with other devices through SPI (Serial Peripheral Interface) communication protocol.

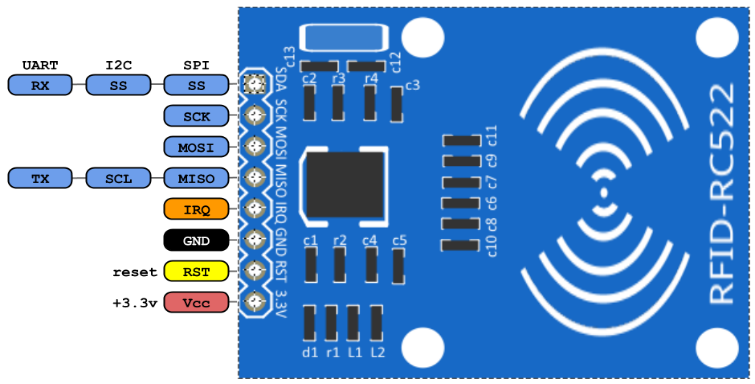


Fig12: MFRC522 showing pin configuration.

Here's a breakdown of the relevant connections on the MFRC522 module:

* **SPI Pins:**
  + **SCK (Serial Clock):** This pin provides the clock signal that synchronizes the data transfer between the MFRC522 and the host microcontroller (like Arduino).
  + **MOSI (Master Out, Slave In):** This pin carries the data transmitted from the host microcontroller to the MFRC522.
  + **MISO (Master In, Slave Out):** This pin carries the data transmitted from the MFRC522 to the host microcontroller.
  + **SS (Slave Select):** This pin is used by the host microcontroller to select the MFRC522 for communication, essentially telling it to listen.
* **Additional Pins:**
  + **IRQ (Interrupt Request):** This pin can be used by the MFRC522 to signal the host microcontroller about specific events, like detecting a tag. This is optional for most basic setups.
  + **RST (Reset):** This pin resets the MFRC522 module.
  + **3.3V and GND:** These pins provide power (3.3V) and ground connection for the MFRC522 module.
  + **Antenna:** The antenna is connected to a dedicated port on the MFRC522 for transmitting and receiving radio signals to communicate with RFID tags.

**3.3 Accelerometer ADXL345**

The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface.

The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (3.9 mg/LSB) enables measurement of inclination changes less than 1.0°.

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion by comparing the acceleration on any axis with user-set thresholds. Tap sensing detects single and double taps in any direction. Free-fall sensing detects if the device is falling. These functions can be mapped individually to either of two interrupt output pins. An integrated memory management system with a 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor activity and lower overall system power consumption.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

The ADXL345 is supplied in a small, thin, 3 mm × 5 mm × 1 mm, 14-lead, plastic package.

**Functional block diagram**

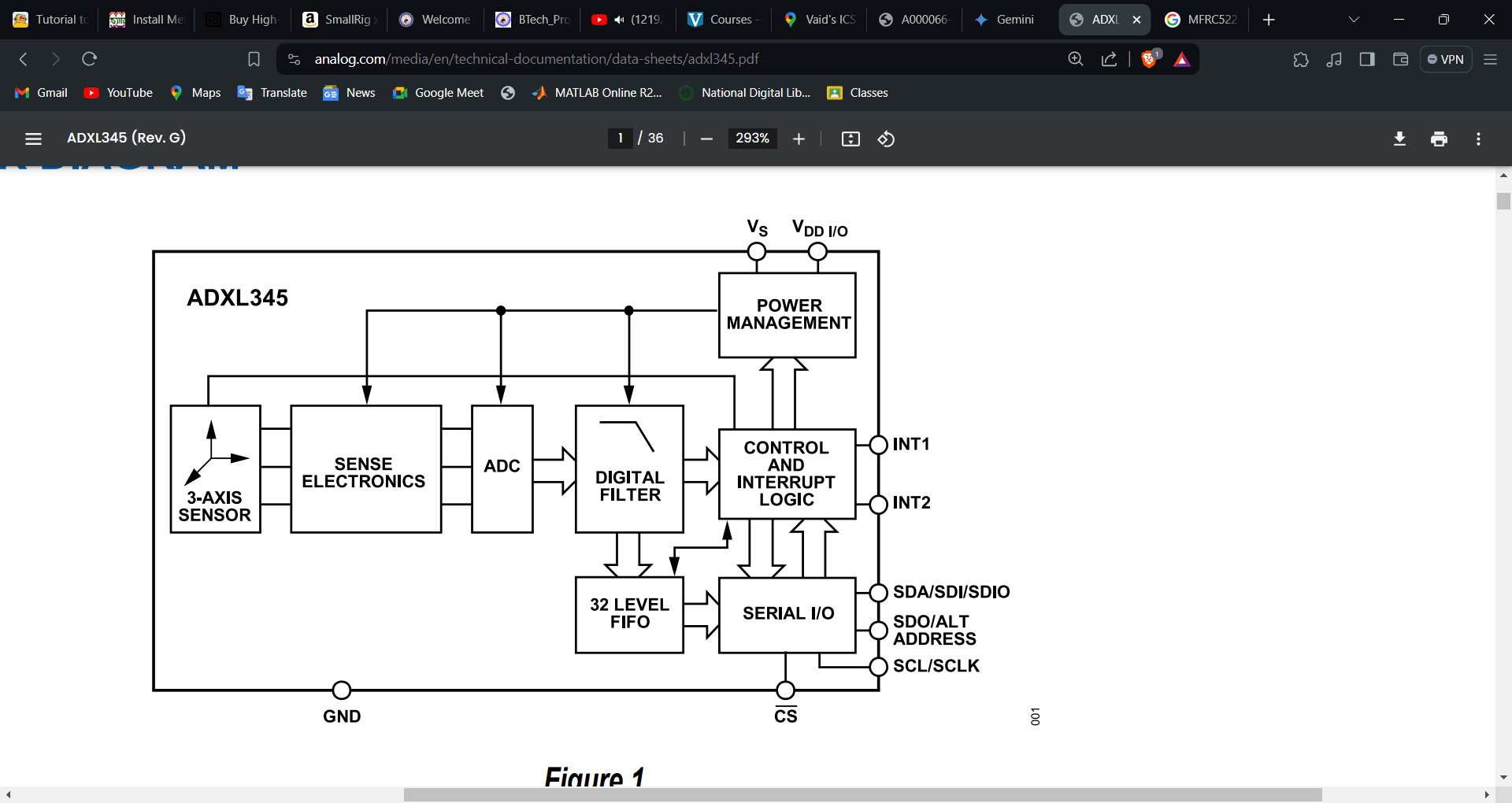
****

Fig13: Simplified block diagram of ADXL345.

**Pinning information of ADXL345**

The ADXL345 is a 3-axis accelerometer that comes in a small, 14-pin package.

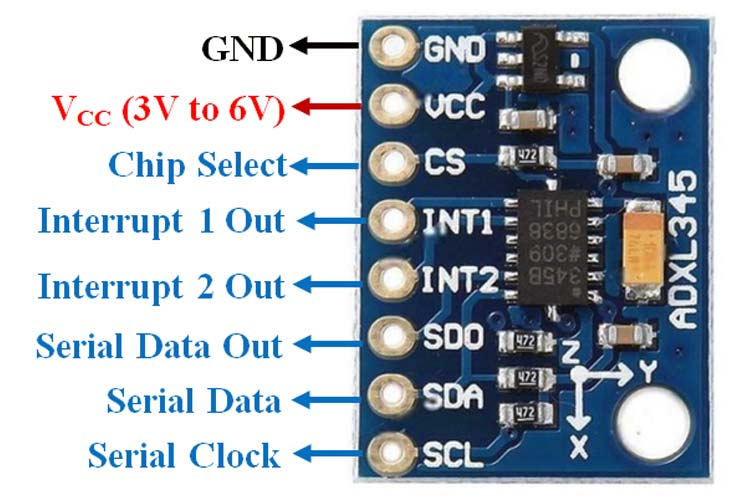


Fig14: ADXL345 showing various pin.

Here's a breakdown of the important ADXL345 pins:

**Power and Ground:**

* **VCC:** This pin supplies power to the ADXL345, typically 3.3V.
* **GND:** This pin is the ground connection for the ADXL345.

**Communication Interfaces:**

The ADXL345 can communicate with your microcontroller through two different interfaces: SPI (Serial Peripheral Interface) or I2C. You'll need to choose one based on your project's needs and microcontroller capabilities.

* **SPI Pins (if using SPI mode):**
  + **SCK (Serial Clock):** This pin provides the clock signal that synchronizes data transfer between the ADXL345 and the microcontroller.
  + **MOSI (Master Out, Slave In):** This pin carries the data transmitted from the microcontroller to the ADXL345.
  + **MISO (Master In, Slave Out):** This pin carries the data transmitted from the ADXL345 to the microcontroller. (Optional, not required for all SPI configurations)
  + **CS (Chip Select):** This pin is used by the microcontroller to select the ADXL345 for communication.
* **I2C Pins (if using I2C mode):**
  + **SCL (Serial Clock):** This pin provides the clock signal for the I2C communication protocol.
  + **SDA (Serial Data):** This pin carries the data between the ADXL345 and the microcontroller on the I2C bus.

**Other Important Pins:**

* **INT1 and INT2 (Interrupt):** These pins can be used to trigger interrupts on the microcontroller based on specific events from the ADXL345, such as detecting taps or freefall.
* **SDO/ALT ADDR (Selectable Data Output/Alternate Address):** This pin has a dual function. In SPI mode, it can be used as an additional data output (MISO) pin. In I2C mode, it can be used to set the I2C slave address of the ADXL345 (optional).

**3.4 Sim808 (GPS&GSM module)**

SIM808 is integrated with a high performance GSM/GPRS engine, a GPS engine and a BT engine. The GSM/GPRS engine is a quad-band GSM/GPRS module that works on frequencies GSM 850MHz, EGSM 900MHz, DCS 1800MHz and PCS 1900MHz. SIM808 features GPRS multi-slot class 12/ class 10 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4. The GPS solution offers best- in-class acquisition and tracing sensitivity, Time-To-First-Fix (TTFF) and accuracy.

With a tiny configuration of 24\*24\*2.6mm, SIM808 can meet almost all the space requirements in user applications, such as M2M, smart phone, PDA, tracker and other mobile devices.

SIM808 has 68 SMT pads, and provides all hardware interfaces between the module and customers’ boards.

* Support 4\*4\*2 keypads.
* One full modem serial port.
* One USB, the USB interfaces can debug, download software.
* Audio channels which include a microphone input and a receiver output.
* One SIM card interface.
* Charging interface.
* Programmable general purpose input and output.
* Support Bluetooth function.
* Support PWM and ADC.
* PCM/SPI/SD card interface, only one function can be accessed synchronously. (Default function is PCM).

SIM808 is designed with power saving technique so that the current consumption is as low as 1mA in sleep mode (GPS engine is powered down).

SIM808 integrates TCP/IP protocol and extended TCP/IP AT commands which are very useful for data transfer applications.

SIM808 Functional Diagram

The following figure shows a functional diagram of SIM808:

z The GSM baseband engine

z The GPS engine

z Flash

z The GSM radio frequency part

z The antenna interface

z The other interfaces

**SIM808 Functional Diagram**

The following figure shows a functional diagram of SIM808:

* The GSM baseband engine
* The GPS engine
* Flash
* The GSM radio frequency part
* The antenna interface
* The other interfaces

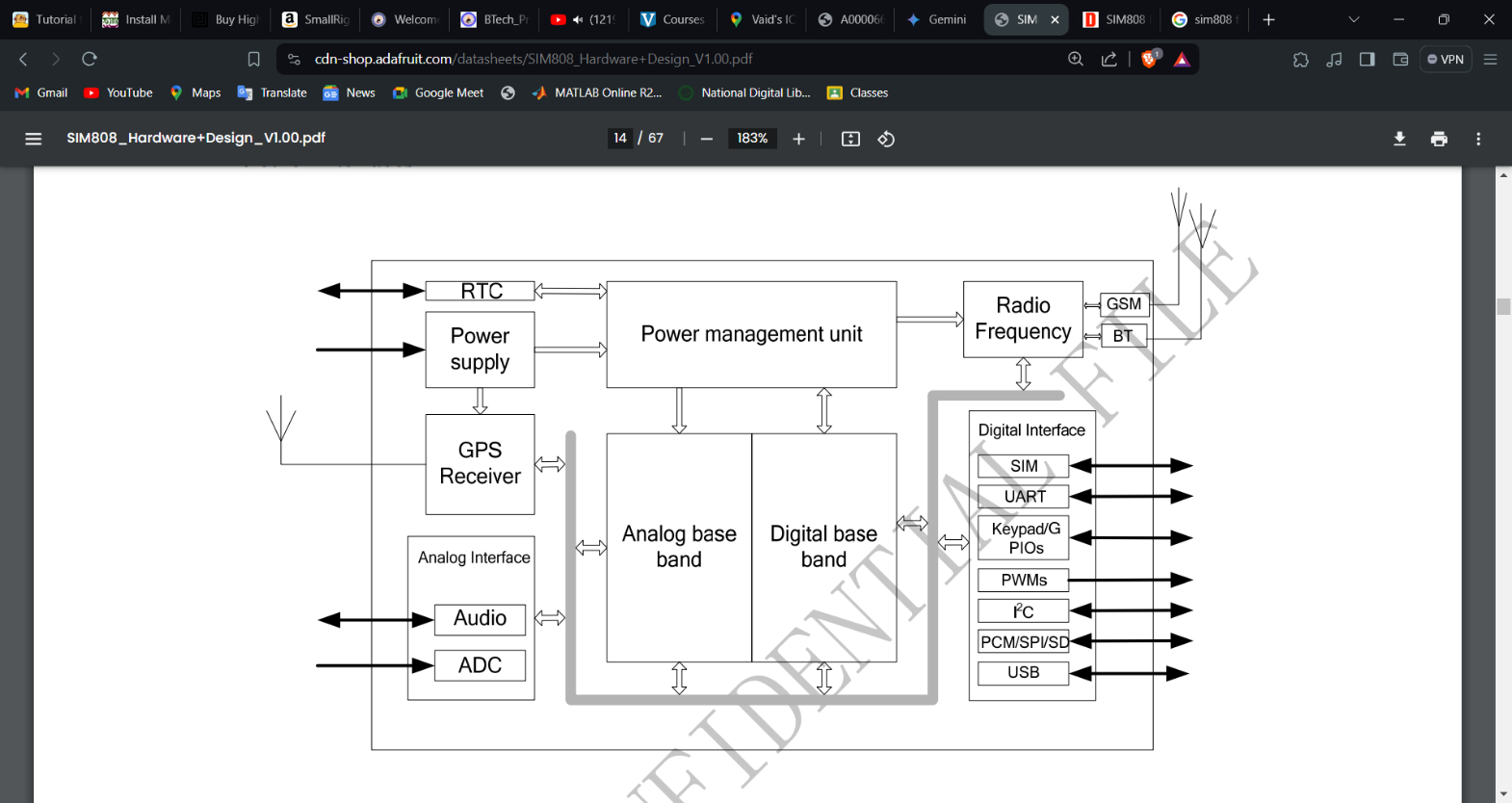


Fig15: Functional block diagram of Sim808.

**Pinning information of Sim808**

The SIM808 GSM GPRS module has a variety of pins for power supply, communication, control, and functionality.

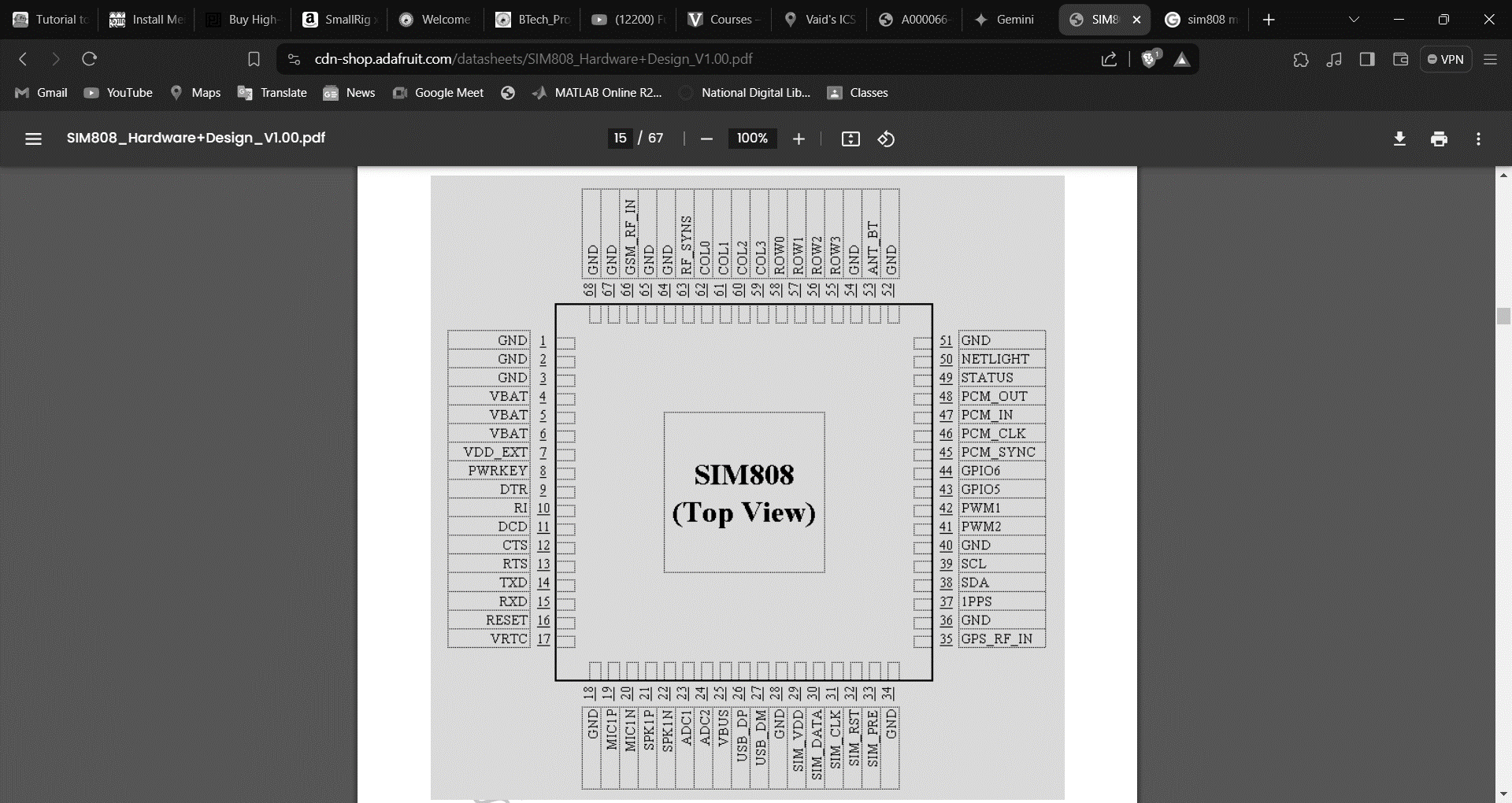
****

Fig16: Pin diagram of Sim808.

Here's a breakdown of the important SIM808 pins:

**Power Supply:**

* **VBAT:** Main power supply pin for the SIM808 module. Voltage range is typically between 3.4V and 4.4V.
* **GND:** Ground connection for the module.

**SIM Card Interface:**

* **SIM\_DETECT:** This pin outputs a logic signal indicating the presence of a SIM card.
* **SIM\_POWER:** Controls the power supply to the SIM card.

**Communication Interfaces:**

* **UART Pins (TXD, RXD):** These pins are used for serial communication between the SIM808 and your microcontroller using UART protocol.
  + **TXD (Transmit Data):** Transmits data from the SIM808 to the microcontroller.
  + **RXD (Receive Data):** Receives data from the microcontroller to the SIM808.
* **USB Port:** This port is primarily used for firmware updates on the SIM808 module and not for general communication.

**Control and Functionality Pins:**

* **PWRKEY:** This pin can be used to power on or reset the SIM808 module.
* **MIC:** Microphone input for audio recording.
* **SPK:** Speaker output for audio playback.
* **STATUS:** This pin outputs a logic signal that can indicate different module statuses.

**Additional Pins:**

* **ADC:** Some SIM808 modules might have additional Analog-to-Digital Converter (ADC) pins that can be used to read analog voltage signals from external sensors.
* **I2C:** Some variants might have dedicated I2C communication pins for connecting with I2C sensors.

**3.5 5V RELAY**

A 5V relay module is a compact and convenient device that allows you to control high voltage or high current appliances using a low voltage signal from a microcontroller like Arduino. It acts like an automatic switch, bridging the gap between the low-power control world and high-power devices.

The key aspects of a 5V relay module:

* **Function:** It essentially works as an electromagnetic switch. When a low voltage (typically 5V DC) is applied to the control input of the module, it activates an electromagnet within the relay. This electromagnet attracts a metal armature, which physically switches the contacts of the relay, turning on or off the connected device in the high voltage/current circuit.
* **Components:** A 5V relay module typically consists of two main parts:
  + **Relay:** This is the core electromagnetic switching component. It has a coil, an armature, and contacts.
  + **Control Board:** This circuit board provides the connection points for the control signal (usually 5V), power supply (often 5V DC), and the high voltage/current terminals to control external devices.
* **Applications:** 5V relay modules are incredibly versatile and can be used in various projects due to their ability to handle high power with a low-power control signal. Here are some common applications:
  + **Home Automation:** Turning on/off lights, appliances, or controlling motors for things like automated curtains or pet feeders.
  + **Robotics:** Controlling various components within a robot, like powering motors or solenoids.
  + **Data Acquisition Systems:** Switching sensors or actuators based on control signals.
  + **Internet of Things (IoT) Projects:** Enabling remote control of devices through Wi-Fi or Bluetooth modules.
* **Benefits:**
  + **Safe Control:** Allows you to control high voltage/current devices using a low voltage signal, keeping the control circuit safer from potential hazards.
  + **Easy Integration:** Compact size and simple connection requirements make them easy to integrate into various projects.
  + **Versatility:** Their ability to handle a wide range of voltages and currents makes them suitable for diverse applications.
* **Considerations:** When choosing a 5V relay module, it’s important to consider the maximum voltage and current ratings it can handle for the load you want to control. Also, make sure the module has the required number of channels (number of independent relays) for your project.

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **Description** |
| 1 | Coil End 1 | Used to trigger(On/Off) the Relay, Normally one end is connected to 5V and the other end to ground |
| 2 | Coil End 2 | Used to trigger(On/Off) the Relay, Normally one end is connected to 5V and the other end to ground |
| 3 | Common (COM) | Common is connected to one End of the Load that is to be controlled |
| 4 | Normally Close (NC) | The other end of the load is either connected to NO or NC. If connected to NC the load remains connected before trigger |
| 5 | Normally Open (NO) | The other end of the load is either connected to NO or NC. If connected to NO the load remains disconnected before trigger |

Table1: Relay pin configuration.

**Features of 5-Pin 5V Relay**

* Trigger Voltage (Voltage across coil) : 5V DC
* Trigger Current (Nominal current) : 70Ma
* Maximum AC load current: 10A @ 250/125V AC
* Maximum DC load current: 10A @ 30/28V DC
* Compact 5-pin configuration with plastic moulding
* Operating time: 10msec Release time: 5msec
* Maximum switching: 300 operating/minute (mechanically)

**Applications of Relay**

* Commonly used in switching circuits.
* For Home Automation projects to switch AC loads
* To Control (On/Off) Heavy loads at a pre-determined time/condition
* Used in safety circuits to disconnect the load from supply in event of failure
* Used in Automobiles electronics for controlling indicators glass motors etc.

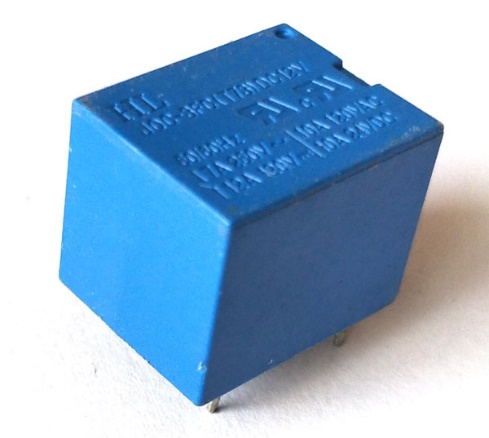


Fig17: 5V Relay module.

**3.6 JUMPER WIRES**

Jumper wires are your best friends when it comes to prototyping circuits with Arduino and other electronic components on breadboards. They are essentially pre-made wires with male connector pins on each end that allow you to easily connect various components without soldering.



Fig18: Jumper wires.

The key features of jumper wires for Arduino:

* **Function:** They simplify circuit creation on breadboards by providing a quick and reusable way to establish connections between components.
* **Components:**
  + **Wires:** Jumper wires come in various lengths and colors. The most common type is solid-core wire, which is more rigid but holds its shape well for breadboard connections.
  + **Male Connector Pins:** Each end of the wire has a male connector pin that fits snugly into the holes of a breadboard. These pins allow for easy insertion and removal without damaging the breadboard.
* **Benefits:**
  + **Breadboard Friendly:** Designed specifically for use with breadboards, making temporary connections a breeze.
  + **Reusable:** Jumper wires can be easily disconnected and reused in different circuits, reducing waste and saving time.
  + **Variety:** Available in various lengths and colors, which can help with project organization and visual clarity when following circuit diagrams. Colors don't have a standardized meaning, but some people use them to differentiate between signal types (e.g., red for power, black for ground).
* **Drawbacks:**
  + **Not Permanent:** Jumper wires are ideal for prototyping, but they are not meant for permanent connections in final projects. Soldering is generally preferred for long-term durability.
  + **Can Get Messy:** With complex circuits, using a large number of jumper wires can lead to a cluttered breadboard, making it difficult to troubleshoot or follow connections.
* **Tips for Using Jumper Wires:**
  + Choose the right length for your connections to avoid unnecessary clutter.
  + Consider using different colored wires to improve code readability and identify different signal types (optional).
  + Keep your breadboard organized to make troubleshooting easier.
  + For final projects, transition from jumper wires to soldered connections for a more robust and permanent solution.

**CHAPTER 4**

**HARDWARE AND SOFTWARE INTEGRATION**

**Step by step for implementation:**

* 1. Use RFID module to check the matching of the tags.
  2. If the tags is matched, it goes further to check.
  3. If the tags is not matched, ignition line id cut off.
  4. If vehicular movement is detected, it will send sms and call alert to the owner.
  5. If vehicular movement is not detected, it will further check for tags.

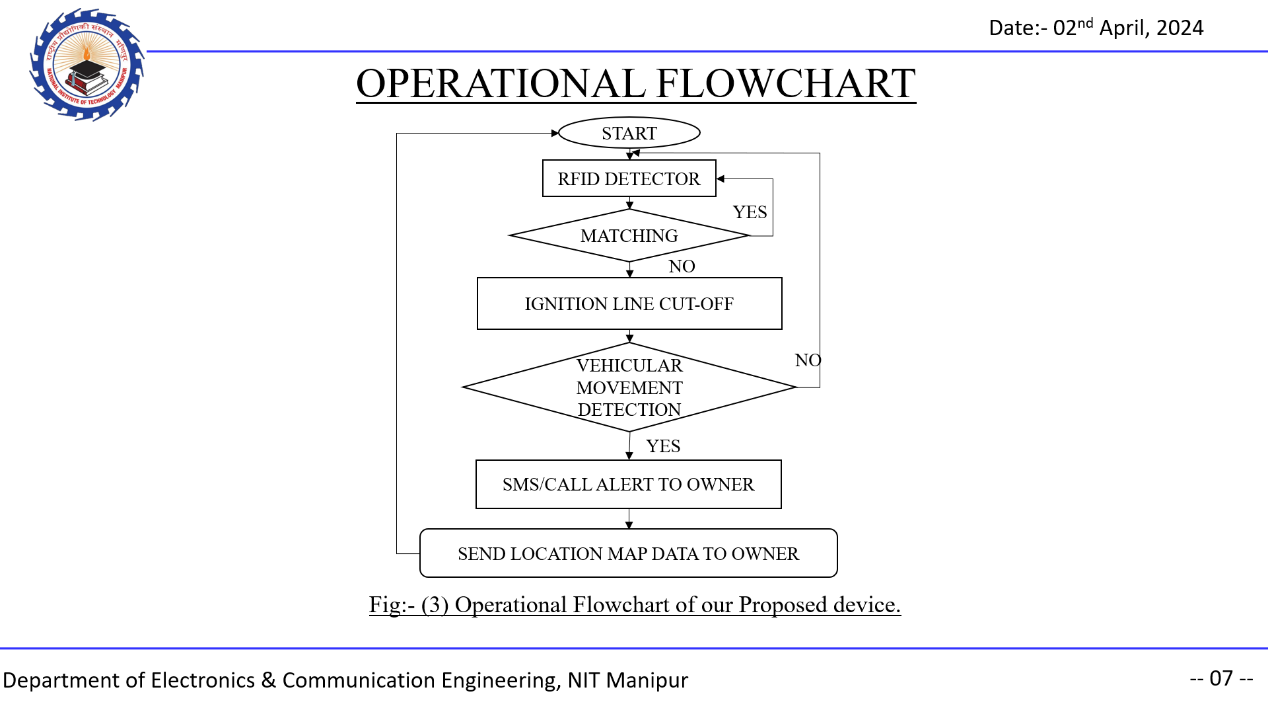
****

Fig19: Operational Flowchart of our Proposed device.

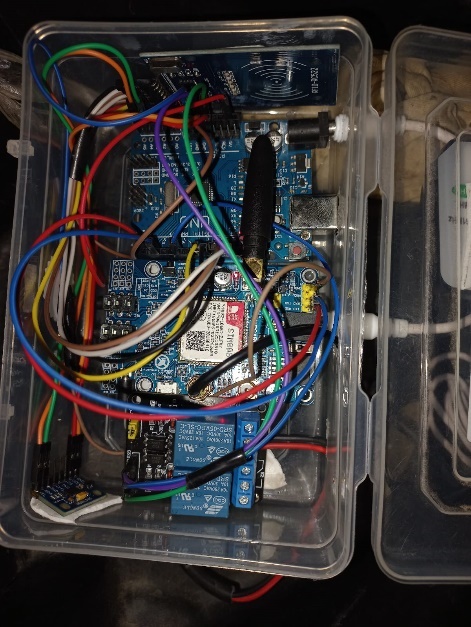


Fig20: Implementation of our device on a vehicle.

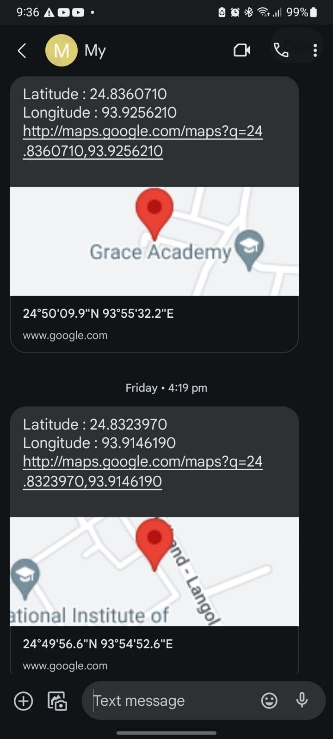


Fig21: Sample picture of device sending location.

****

Fig22: Sample picture of device calling.

**CHAPTER 5**

**CONCLUSION**

The development and implementation of the proposed anti-theft device for vehicles represent a significant stride towards enhancing security and reducing the risk of vehicle theft. By integrating RFID, accelerometer, GPS, and GSM technologies, the device offers a multi-layered approach to safeguarding vehicles, ensuring both reliability and cost-effectiveness.

Through rigorous testing on a two-wheeler scooter and subsequent positive feedback, the effectiveness and practicality of the device have been demonstrated in real-world scenarios. Its ability to provide multiple levels of security, including authentication, movement detection, and real-time tracking, offers a comprehensive solution to address the evolving challenges of vehicle theft.

Furthermore, the success of this project not only benefits vehicle owners by providing peace of mind and protection for their assets but also serves as an inspiration for future developers to innovate in the field of vehicle security. By sharing our experiences and insights, we aim to foster a community of collaboration and innovation, driving further advancements in technology aimed at mitigating the risk of vehicle theft and loss.

In essence, the proposed anti-theft device stands as a testament to the power of technological innovation in addressing societal challenges. Its potential to enhance security, coupled with its practicality and effectiveness, positions it as a valuable asset in the ongoing efforts to safeguard vehicles and promote overall safety on the roads.

**REFERENCES**

[1] **Ashwini Dilip Lahire**, “GPS & GSM based vehicle tracking and security system,” International Journal of Engineering Research and Development, Volume 12, Issue 6, PP.55-60 ,June 2016.

[2] **V.Mulge, Mrs G. Sathyaprabha, Mr P.V.VaraPrasad Rao**, “Anti-Theft Security System Using GSM, GPS, RFID Technology Based On Arm7,” International Journal of Research in Information Technology(IJRIT), Volume 2, Issue 9,Pg. 764-769, September 2014.

[3] **A.Rajasekhar reddy, P.Anwar basha** ,“The Terminal System Design based on hybrid RFID-GPS in Vehicular communications,” International Journal of Modern Engineering Research (IJMER), Vol.2, Issue.4, pp-2316-2319, July-Aug 2012.

[4] **Finkenzeller and R. Handbook**,“Radio-frequency identification fundamentals and applications,” Chippenham: John Wiley & Son, 1999.

[5] **S. Shepard**, RFID: radio frequency identification. McGraw Hill Professional, 2005.

[6] Davis. B, DeLong. R, "Combined remote key control and immobilization system for vehicle security," Power Electronics in Transportation, 1996. IEEE 24-25 Oct. 1996

[7] Khangura, K.S;Middleton,N.V; Ollivier,M.M, "Vehicle anti-theft system uses radio frequency identification," Vehicle Security Systems, IEE Colloquium 8 Oct 1993

[8] Xin Chen, Patrick J, Flynn Kevin.Bowyer, "PCA-based face recognition in infrared imagery: baseline and comparative studies", AMFG 200,. IEEE International Workshop on 17 Oct. 2003

[9] M. Turk, A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, 3(1), pp71 -86, 1991.

[10] Pentland, B. Moghaddam, T. Starner, "View-Based and Modular Eigenspaces for Face Recognition", IEEE CVPR, 1994.

[11] P.Belhumeur, J. Hespanha, and D. Kriegman, " Eigenfaces vs Fisherfaces: Recognition Using Class Specific Linear Projection," IEEE Trans. PAMI 19(7), 1997.

[12] A. Leonardis and H. Bischof H, "Dealing with occlusions in the eigenspace approach," CVPR, pp 453 -458, 1996.

[13] Hong Qiao, Shaoyan Zhang, Bo Zhang and Keane. J, " Face recognition using SVM decomposition methods," IROS 200.

[14] rong-ben,Guo Ke-you and Shi Shu-ming, "A monitoring method of driver fatigue behavior based on machine vision", Intelligent Vehicles Symposium, 2003. Proceedings. IEEE 9-11 June 2003 pp 110-113.

[15] Yinan Ma, Jing Wu, Chengnian Long, Yi-Bing Lin, "MobiDIV: A Privacy-Aware Real-Time Driver Identity Verification on Mobile Phone", IEEE Internet of Things Journal, vol.9, no.4, pp.2802-2816, 2022.

[16] Debashis Das, Sourav Banerjee, Utpal Biswas, "A secure vehicle theft detection framework using Blockchain and smart contract", Peer-to-Peer Networking and Applications, vol.14, no.2, pp.672, 2021.

[17] Debashis Das, Sourav Banerjee, Uttam Ghosh, Utpal Biswas, Ali Kashif Bashir, "A decentralized vehicle anti-theft system using Blockchain and smart contracts", Peer-to-Peer Networking and Applications, vol.14, no.5, pp.2775, 2021.

**APPENDIX**

1. **Arduino code for implementing the device**

#include <SPI.h>

#include <MFRC522.h>

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_ADXL345\_U.h>

#include <SoftwareSerial.h>

#include <DFRobot\_SIM808.h>

#define RST\_PIN 9

#define SS\_PIN 10

#define RELAY\_PIN 6

#define PIN\_TX    2

#define PIN\_RX    3

#define PHONE\_NUMBER  "+91xxxxxxxxxx"

char lat[12];

char lon[12];

char \*gpsLoc;

char phone[16];

SoftwareSerial mySerial(PIN\_TX, PIN\_RX);

DFRobot\_SIM808 sim808(&mySerial);

MFRC522 mfrc522(SS\_PIN, RST\_PIN);

Adafruit\_ADXL345\_Unified accel = Adafruit\_ADXL345\_Unified(12345);

byte authorizedUID[] = {0xB3, 0x8C, 0x8E, 0xA6};

bool isAuthenticated = false;

unsigned long lastAccessTime = 0;

const unsigned long accessTimeout = 5000;

void setup() {

  mySerial.begin(9600);

  Serial.begin(9600);

  SPI.begin();

  mfrc522.PCD\_Init();

  pinMode(RELAY\_PIN, OUTPUT);

  digitalWrite(RELAY\_PIN, HIGH);

  Serial.println(F("Present card for access control"));

  if (!accel.begin()) {

    Serial.println("Ooops, no ADXL345 detected ... Check your wiring!");

    while (1);

  }

  accel.setRange(ADXL345\_RANGE\_16\_G);

  Serial.println("ADXL345 ready!");

  while (!sim808.init()) {

    delay(1000);

    Serial.println("Sim808 init error");

  }

  Serial.println("Sim808 init success");

  Serial.println("Start to call ...");

  if (sim808.attachGPS())

    Serial.println("Open the GPS power success");

  else

    Serial.println("Open the GPS power failure");

  Serial.println("Init Success, please send SMS message to me!");

}

void loop() {

  bool access\_granted = false;

  if (!mfrc522.PICC\_IsNewCardPresent() || !mfrc522.PICC\_ReadCardSerial()) {

    if (!isAuthenticated || millis() - lastAccessTime >= accessTimeout) {

      isAuthenticated = false;

      digitalWrite(RELAY\_PIN, HIGH);

      Serial.println(F("Authorization revoked due to timeout"));

      access\_granted = false;

      mfrc522.PICC\_HaltA();

      mfrc522.PCD\_StopCrypto1();

    }

    delay(50);

  }

  else {

    if (compareUID(mfrc522.uid.uidByte, authorizedUID, mfrc522.uid.size)) {

      isAuthenticated = true;

      lastAccessTime = millis();

      Serial.println(F("Access Granted."));

      digitalWrite(RELAY\_PIN, LOW);

      return;

    }

    else {

      isAuthenticated = false;

      Serial.println(F("Access Denied! Present a proper CARD"));

      digitalWrite(RELAY\_PIN, HIGH);

    }

  }

  if (!isAuthenticated) {

    Serial.println("Out of scope");

    float threshold = 11;

    if (isMotionDetected(threshold)) {

      Serial.println("Motion detected!");

        char\* gpsLoc = getGPSLocation();

        if (gpsLoc != NULL) {

          sim808.sendSMS((char\*)PHONE\_NUMBER, gpsLoc);

          sim808.callUp((char\*)PHONE\_NUMBER);

          free(gpsLoc); // Free the dynamically allocated memory

        } else {

          Serial.println("Error: GPS location is NULL.");

        }

      }

    }

  checkSMS();

}

bool compareUID(byte\* uid1, byte\* uid2, byte size) {

  for (byte i = 0; i < size; i++) {

    if (uid1[i] != uid2[i]) {

      return false;

    }

  }

  return true;

}

bool isMotionDetected(float threshold) {

  sensors\_event\_t event;

  accel.getEvent(&event);

  float x = event.acceleration.x;

  float y = event.acceleration.y;

  float z = event.acceleration.z;

  float magnitude = sqrt(x \* x + y \* y + z \* z);

  Serial.println(magnitude);

  return magnitude > threshold;

}

 void checkSMS() {

  int messageIndex = sim808.isSMSunread();

  if (messageIndex > 0) {

    char datetime[24];

    char sender[16];

    char message[200];

    // Read the SMS message

    if (sim808.readSMS(messageIndex, message, 200, sender, datetime)) {

      Serial.println("Msg received from: ");

      Serial.println(sender);

      Serial.println("Message content: ");

      Serial.println(message);

      String msgContent = String(message);

      msgContent.toLowerCase();

      if (msgContent.indexOf("gps") != -1) {

      char\* gpsLoc = getGPSLocation();

      if (gpsLoc != NULL) {

          sim808.sendSMS(sender, gpsLoc);

          free(gpsLoc); // Free the dynamically allocated memory

        } else {

          Serial.println("Error: GPS location is NULL.");

        }

      }

    }

  }

}

char\* getGPSLocation() {

  // Keep trying until GPS data is available

  while (!sim808.getGPS()) {}

  // Convert latitude and longitude to strings

  char lat[12];

  char lon[12];

  dtostrf(sim808.GPSdata.lat, 6, 7, lat);

  dtostrf(sim808.GPSdata.lon, 6, 7, lon);

  // Construct GPS location string

  char\* gpsLoc = (char\*)malloc(100);

  if (gpsLoc != NULL) {

    snprintf(gpsLoc, 100, "Latitude : %s\nLongitude : %s\nhttp://maps.google.com/maps?q=%s,%s\n", lat, lon, lat, lon);

  } else {

    Serial.println("Error: Failed to allocate memory for GPS location.");

  }

  return gpsLoc;

}