Project Proposal

EC6020: Embedded Systems Design

Presented to

Nishankar sathiyamohan

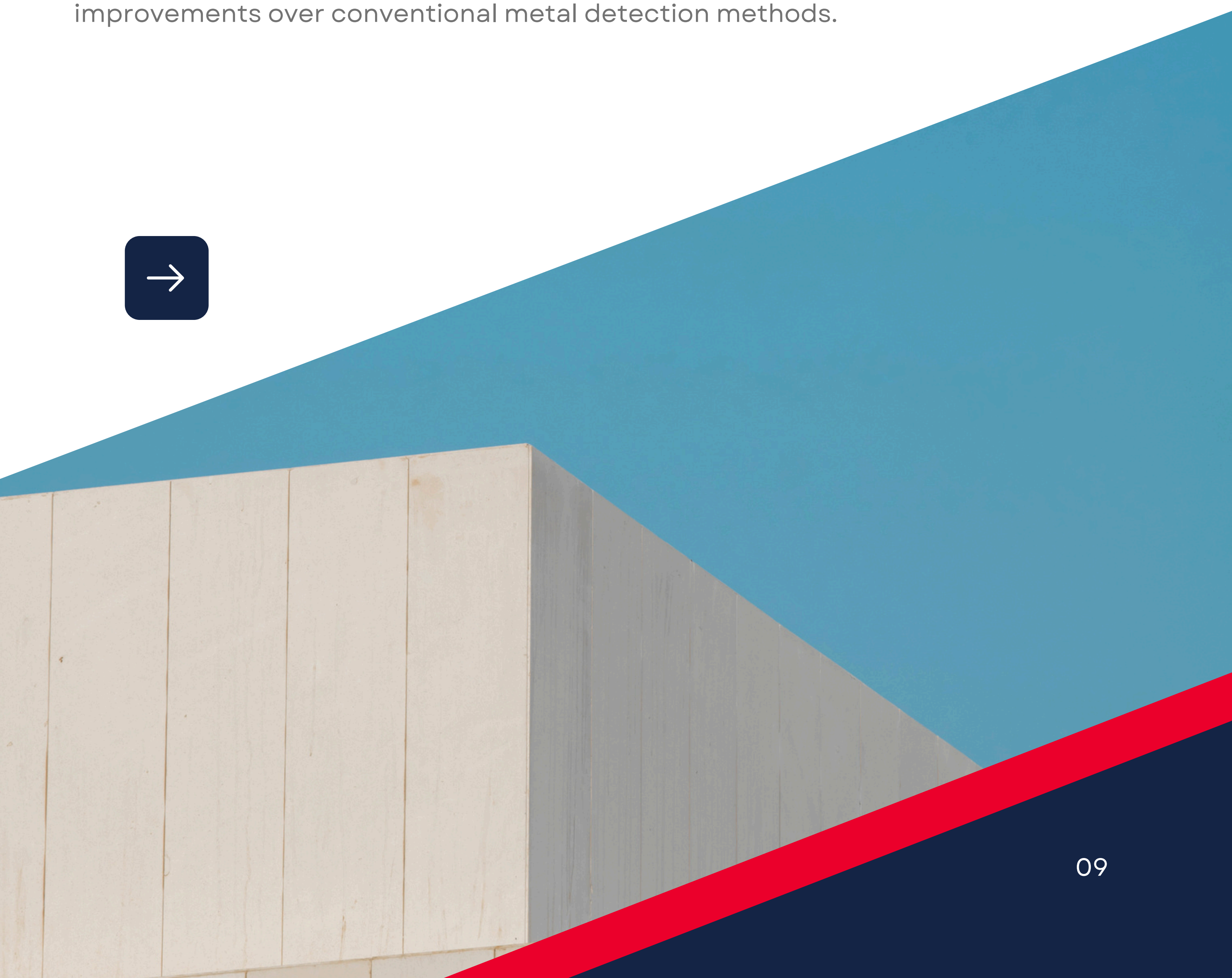
FACULTY OF ENGINEERING
UNIVERSITY OF JAFFNA

Agenda

Introduction

Smart Metal Detector

Metal detection technology has been widely used in security, industrial, and archaeological applications. However, traditional metal detectors often lack intelligent capabilities, requiring manual operation and interpretation. Our project, Smart Metal Detector, aims to integrate modern sensor technology and IoT connectivity to enhance metal detection accuracy, efficiency, and usability.
This smart system will utilize advanced sensors to differentiate between various types of metals, reducing false alarms. Additionally, it will incorporate real-time data processing and wireless connectivity to enable remote monitoring and logging. By integrating the proposed system will provide a user-friendly and efficient solution for security inspections, industrial applications, and environmental safety.
This project will focus on designing and developing a prototype that demonstrates the enhanced capabilities of a smart metal detector, offering significant improvements over conventional metal detection methods.



About Us – The Team

Amarasooriya K.A.C.T.

2021/E/079

Gamage M.G.C.T.

2021/E/086

Ariyarathna K.T.S.P.

2021/E/068

Wijesooriya G.G.P.N.

2021/E/040

Silva K.C.D.

2021/E/019

Problem

1. **High False Alarms**
- Traditional metal detectors often struggle to differentiate between various types of metals, leading to frequent false alarms. This reduces their reliability in critical applications such as security screening.

2. **Lack of Smart Classification**
- Most conventional metal detectors only indicate the presence of metal but do not provide details about the type or composition. This makes them less effective in industrial and archaeological applications.

3. **Manual Operation and Limited Automation**
- Traditional systems require continuous human supervision and interpretation. This can be time-consuming and inefficient, especially in large-scale scanning operations.

4. **Limited Detection in Complex Environments**
- Many metal detectors struggle in environments with high electromagnetic interference or challenging terrains, reducing their effectiveness in real-world applications like underwater detection or buried object identification.

5. **No Remote Monitoring and Data Logging**
- Most existing metal detectors do not support IoT-based monitoring or data storage. This lack of connectivity prevents real-time tracking, which is essential for security and industrial safety applications.

6. **Energy Inefficiency**
- Many metal detectors consume excessive power, making them less practical for portable or long-duration use. An energy-efficient solution is required to enhance battery life and usability.

Solution

1. Smart Metal Classification
✅ Solution: Advanced Signal Processing & Material Identification
- Process the signal characteristics (frequency, amplitude, conductivity) to identify different metals.
- Display metal type (e.g., iron, copper, aluminum) on a smartphone interface for easy interpretation.

2. Automation and Hands-Free Operation
✅ Solution: IoT-Based Remote Monitoring & Control
- Connect the metal detector to a smartphone app via Wi-Fi/Bluetooth for real-time monitoring.
- Automate alerts, sensitivity adjustments, and logging to minimize manual operation.

3. Enhanced Detection in Complex Environments
✅ Solution: Adaptive Calibration & Interference Reduction
- Implement automatic sensitivity adjustment to counteract electromagnetic interference in challenging environments.
- Use filtering algorithms to minimize noise and improve metal detection accuracy.

4. Real-Time Data Logging & Remote Access
✅ Solution: Cloud-Based Storage & IoT Dashboard
- Store detection data in a cloud database for historical analysis.
- Users can access detection logs, timestamps, and location tracking through a mobile app or web interface.

5. Energy Efficiency & Portability
✅ Solution: Low-Power Design & Smart Power Management
- Use energy-efficient MCUs like ESP32 or Raspberry Pi for processing and connectivity.
- Implement power-saving modes and rechargeable battery support to improve battery life.

Novelty

- Bluetooth or Wi-Fi to send detection data from the metal detector to the phone.
- A mobile app (Blynk, custom Android app, or a web-based interface) to display real-time detection results.
- Simple UI elements like a green/red indicator or a buzzer notification when metal is found.

→

High-level architecture

Metal Detection Unit

✅ Technology Used: Inductive/Capacitive Sensor or Pulse Induction Sensor

✅ Reason:

- High sensitivity to different types of metals.
- Works well in various environments (e.g., soil, water, industrial settings).
- Provides an electrical signal based on metal detection.

2. Microcontroller/Processing Unit

✅ Technology Used: ESP32 or Arduino with Bluetooth/Wi-Fi Module

✅ Reason:

- ESP32: Built-in Wi-Fi and Bluetooth, low power consumption, and fast processing for IoT integration.
- Arduino with Wi-Fi module (ESP8266): Alternative if using a simpler setup.
- Processes the sensor signal and transmits the detection result to the mobile display.

3. IoT Connectivity

✅ Technology Used: Bluetooth or Wi-Fi Communication

✅ Reason:

- Bluetooth: Low power, ideal for short-range detection (within 10-15m).
- Wi-Fi: Suitable for real-time data logging and cloud storage if remote monitoring is needed.
- Ensures seamless mobile display integration without additional screens.

→

High–level architecture

4. Mobile Display & User Interface
✅ Technology Used: Blynk App / MIT App Inventor / Custom Android App
✅ Reason:

- Blynk App: Easiest for IoT projects, requires minimal coding.
- MIT App Inventor: Drag-and-drop interface for simple UI development.
- Custom Android App: If a more advanced UI and real-time graphing are required.
- Provides a clear visual indication of metal detection (YES/NO or signal strength).

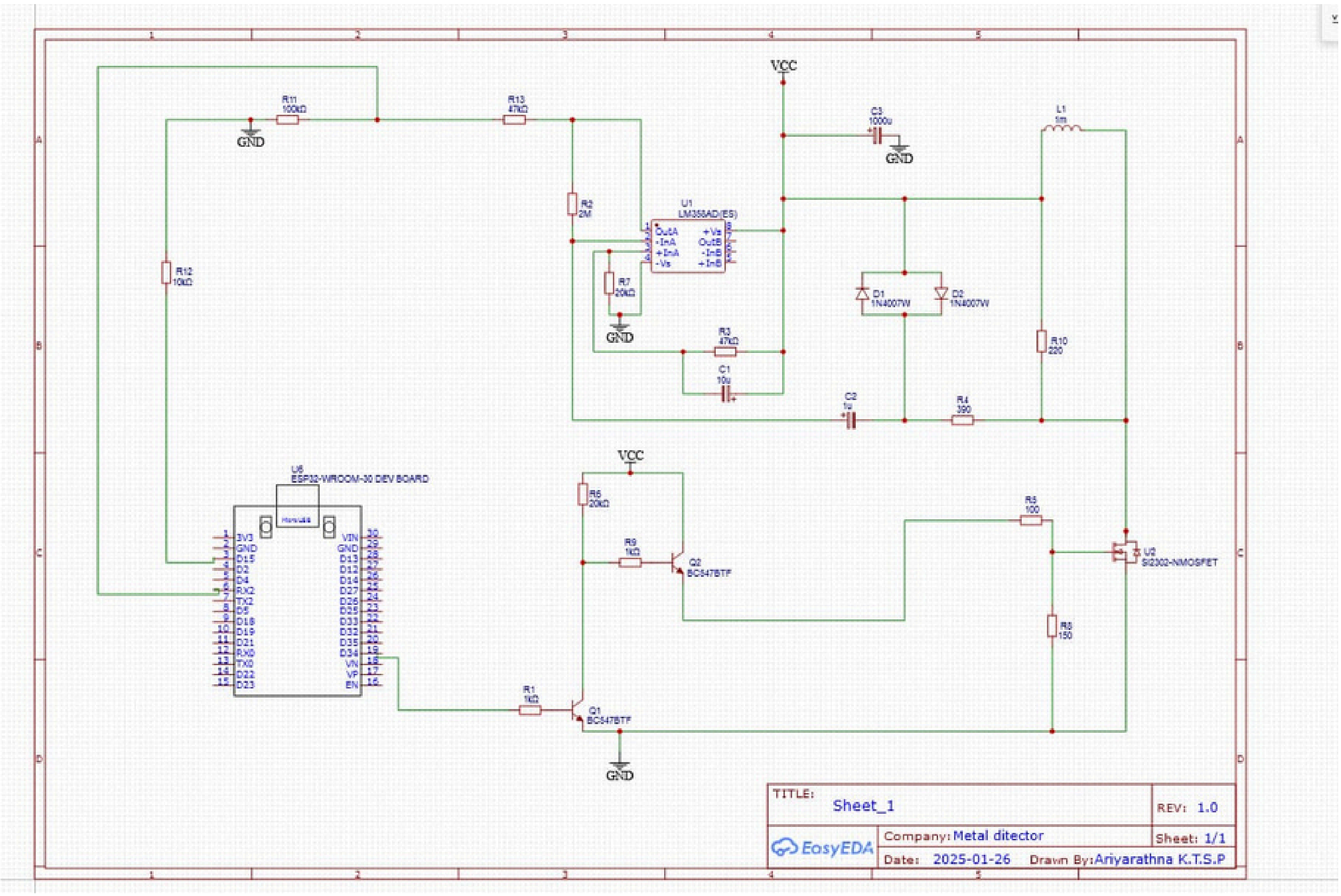
5. Power Supply
✅ Technology Used: Rechargeable Li-ion Battery with Power Management
✅ Reason:

- Provides long-lasting power for field use.
- Can support energy-efficient operation of ESP32 and sensors.

Why This Architecture?
✅ Optimized for IoT – Enables mobile-based metal detection without extra hardware.
✅ Cost-Effective & Scalable – Uses affordable components with room for future upgrades.
✅ User-Friendly – Simple mobile interface eliminates the need for additional displays.
✅ Wireless & Portable – Allows remote monitoring and improves usability in real-world applications.

→

Circuit design and protocols.



Case Project

Budget

Component	Value/Part No.	Quantity	Cost
Microcontroller	ESP32	1	1750
Operational Amplifier	TL072	1	500
Transistor	BC547/BC557	2	300
MOSFET	IRF740	1	500
Resistors	Various(1k,6.8,etc)	~10	50
Capacitors	Various(1mF,1uF,10uF)	~3	50
Diodes	1N4148	2	50

Github repo link

KusalChamu/**Smart-metal-detector**

1 Contributor · 0 Issues · 0 Stars · 0 Forks

KusalChamu/Smart-metal-detector

Contribute to KusalChamu/Smart-metal-detector development by creating an account on GitHub.

GitHub

https://github.com/KusalChamu/Smart-metal-detector.git

PASINDU-040 - Overview

PASINDU-040 has 7 repositories available. Follow their code on GitHub.

GitHub

Timeline

Week 1-2: Project Planning & Research
Define project objectives and scope
Conduct market research and gather requirements
Identify key components and technology stack
Develop a project plan with milestones

Week 3-4: Design Phase
Develop system architecture and data flow diagrams
Finalize materials or software requirements

Week 5-6: Initial Prototyping
Assemble basic hardware or set up software framework
Write initial code for core functionality
Test fundamental operations and iterate on design

Week 7-8: Development & Refinement
Implement main features and functionalities
Integrate components
Conduct internal testing and debugging

Week 9-10: Testing & Optimization
Perform stress tests and refine system performance
Fix bugs and optimize efficiency
Gather feedback from initial testing phase

Week 11: Final Adjustments & Documentation
Implement final improvements based on feedback
Prepare user manual and technical documentation
Conduct a final system test

Week 12: Deployment & Presentation
Deploy the final version

Thank You