IoT DIRfram
BlueVision ITM

COS 301
Capstone Project Proposal
2024

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

IoT, Data traversal, User Interface, Automation

Make today matter

www.up.ac.za

Project Description:

Developing software for a Raspberry Pi that automatically executes when connected via USB or UART to retrieve information from an IoT device. The software will gather details such as firmware version, chip model, and voltage usage, compiling them into an XML format. This information can be stored onboard the Raspberry Pi or transmitted to another system for analysis.

Introduction

Key Features:

- Automatic execution on Raspberry Pi connection to IoT device
- Retrieval of firmware, chip model, and voltage information
- Compilation of data into XML format
- Storage onboard Raspberry Pi or transmission to external system

Technical Stack:

- Raspberry Pi for hardware
- Python for software development
- USB or UART for device communication
- XML for data formatting

Project Goals:

- Provide students with hands-on experience in IoT device communication and data processing
- Demonstrate the use of state-of-the-art SE tools and techniques
- Offer technical challenges within a manageable scope for a team of up to 5 students over 6 months
- Potential for future research in IoT and data analysis

Background & Motivation

Background:

The Internet of Things (IoT) has revolutionized the way devices communicate and interact with each other, leading to a vast array of interconnected devices in various industries. As the number of IoT devices continues to grow, the need for efficient management and monitoring of these devices becomes increasingly important. However, accessing detailed information about IoT devices, such as firmware versions, chip models, and voltage usage, can be challenging, especially in large-scale deployments.

Motivation:

The proposed project aims to address this challenge by developing software for a Raspberry Pi that can automatically retrieve detailed information from IoT devices. This software will not only simplify the process of gathering information, troubleshoot and enable users to store and analyse this data efficiently. By providing a user-friendly interface and supporting both onboard storage and external transmission, the project aims to enhance the usability and scalability of IoT device management systems.

Significance:

This project is significant for several reasons. Firstly, it offers practical experience in developing software for IoT environments, a rapidly growing field with a high demand for skilled professionals. Secondly, it demonstrates the use of state-of-the-art SE tools and techniques, showcasing the students' ability to apply theoretical knowledge to real-world problems. Lastly, the project has the potential to contribute to future scientific research in IoT, software engineering and data analysis, as the gathered information can be used for further analysis and optimization of IoT systems and future software development projects.

System Requirements

Raspberry Pi:
Model: Raspberry Pi 3 or higher
Operating System: Raspbian or compatible Linux distribution
Processor: ARMv7, Qualcomm Krait or higher
Memory: 1GB RAM or higher

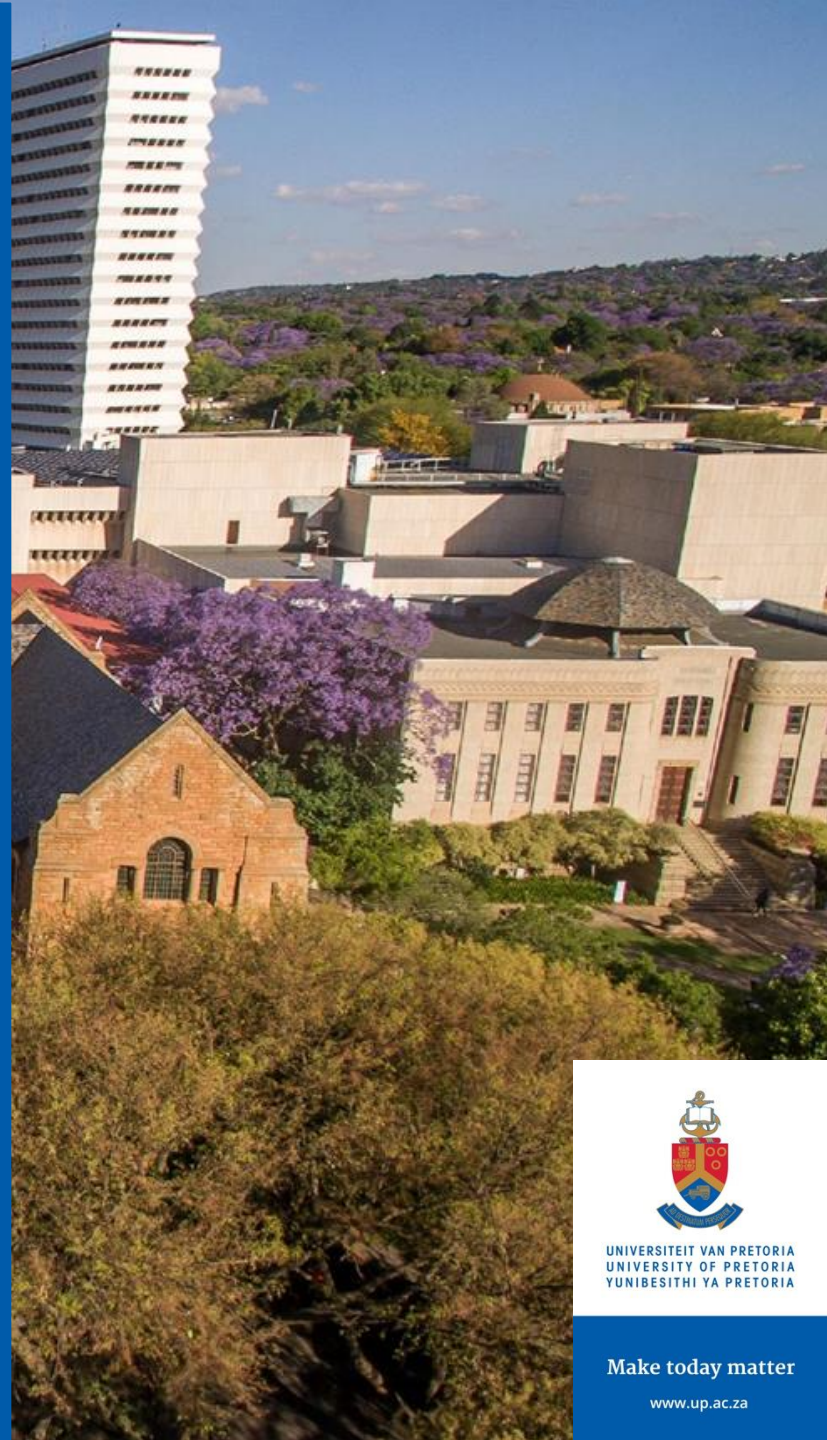
IoT Device:
USB or UART interface for connection to Raspberry Pi
Firmware that supports two-way communication

Software Dependencies:
Python 3.x installed on the Raspberry Pi
Python libraries for USB or UART communication
Python libraries for XML parsing and formatting

Storage (Optional):
MicroSD card for storing retrieved data onboard the Raspberry Pi

External System (Optional):
External system for receiving transmitted data, if not stored onboard (e.g., a computer or server with network connectivity)

Development Environment:
IDE for writing Python scripts (e.g., Visual Studio Code)
SSH or VNC access to the Raspberry Pi for remote development and testing



Core Requirements

Automated Execution:

The software should automatically execute on the Raspberry Pi when connected to an IoT device via USB or UART.

Data Retrieval:

The software should be able to retrieve essential information from the IoT device, Including firmware version, chip model, and voltage usage bonus for any other relevant information gathered.

Data Compilation:

The retrieved information should be compiled into an XML or acceptable format for storage or transmission.

Storage Options:

The software should support storing the compiled data onboard the Raspberry Pi or transmitting it to another system for analysis.

Error Handling:

The software should include robust error handling mechanisms to manage communication errors and other issues that may arise during data retrieval and processing.

Resource Efficiency:

The software should be designed to use system resources efficiently, considering the limited capabilities of the Raspberry Pi.

Wow Factors

1. **Real-Time Monitoring:** Enable real-time monitoring of IoT device metrics, allowing users to track changes and performance over time.
2. **Customizable Reporting:** Provide users with the ability to customize reports and data visualization, enhancing the analysis and presentation of retrieved information.
3. **Remote Configuration:** Allow users to remotely configure the software and IoT devices, providing flexibility and convenience in managing the system.

Architectural Requirements

Modular Design:
The software should be modular, allowing for easy integration of new features and components.

Scalability:
The architecture should be scalable to accommodate a growing number of IoT devices and users.

Reliability:
Ensure the system is reliable, with mechanisms in place to handle errors and failures gracefully.

****Compatibility:**
Ensure compatibility with a wide range of IoT devices and operating systems.

** -Could be considered a core requirement

Constraints

Constraints

1. **Data Validation and Error Checking:** Implement robust data validation and error-checking mechanisms to ensure the accuracy and integrity of retrieved information, especially considering the potential variability and complexity of IoT device data formats.
2. **Cross-Platform Compatibility:** Ensure that the software is compatible with multiple platforms and operating systems, allowing for flexibility in deployment and use across different environments.
3. **Integration with Existing Systems:** Provide integration capabilities with existing IoT management systems or data analysis tools, allowing for seamless interoperability and data exchange.

Budget

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Make today matter

www.up.ac.za

Budget

<Please specify a proposed budget. This includes any components that might have a cost component.>

Project Plan [Optional]

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Make today matter

www.up.ac.za

Team Breakdown [Optional]

<Please provide a proposed team and roles breakdown eg. UI engineer, DevOps, Designer etc.>

Project Plan [Optional]

<Please provide a proposed project plan>

Client Details

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Make today matter

www.up.ac.za

Contact Details

<Please enter industry member contact details here>

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Make today matter

www.up.ac.za

Corporate Assets

<Please provide logos and any corporate identity assets here>

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Make today matter

www.up.ac.za

THANK YOU

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Make today matter

www.up.ac.za