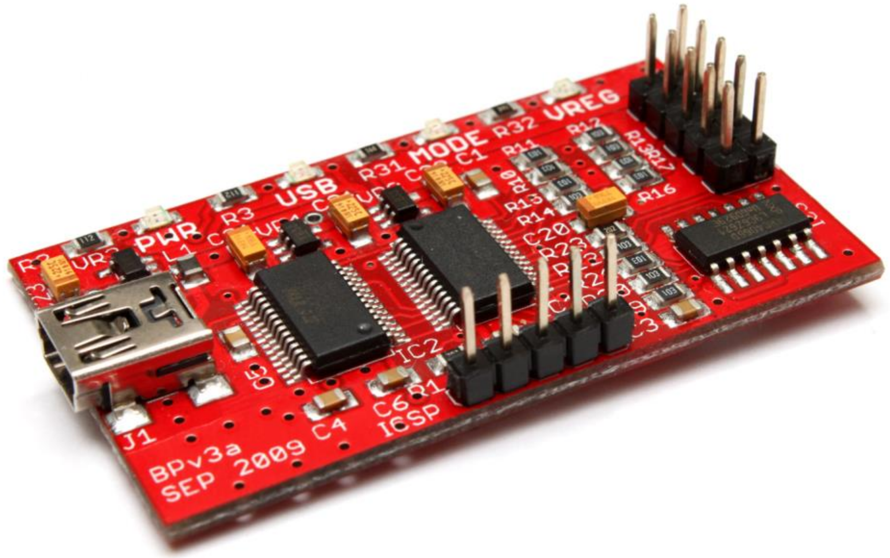
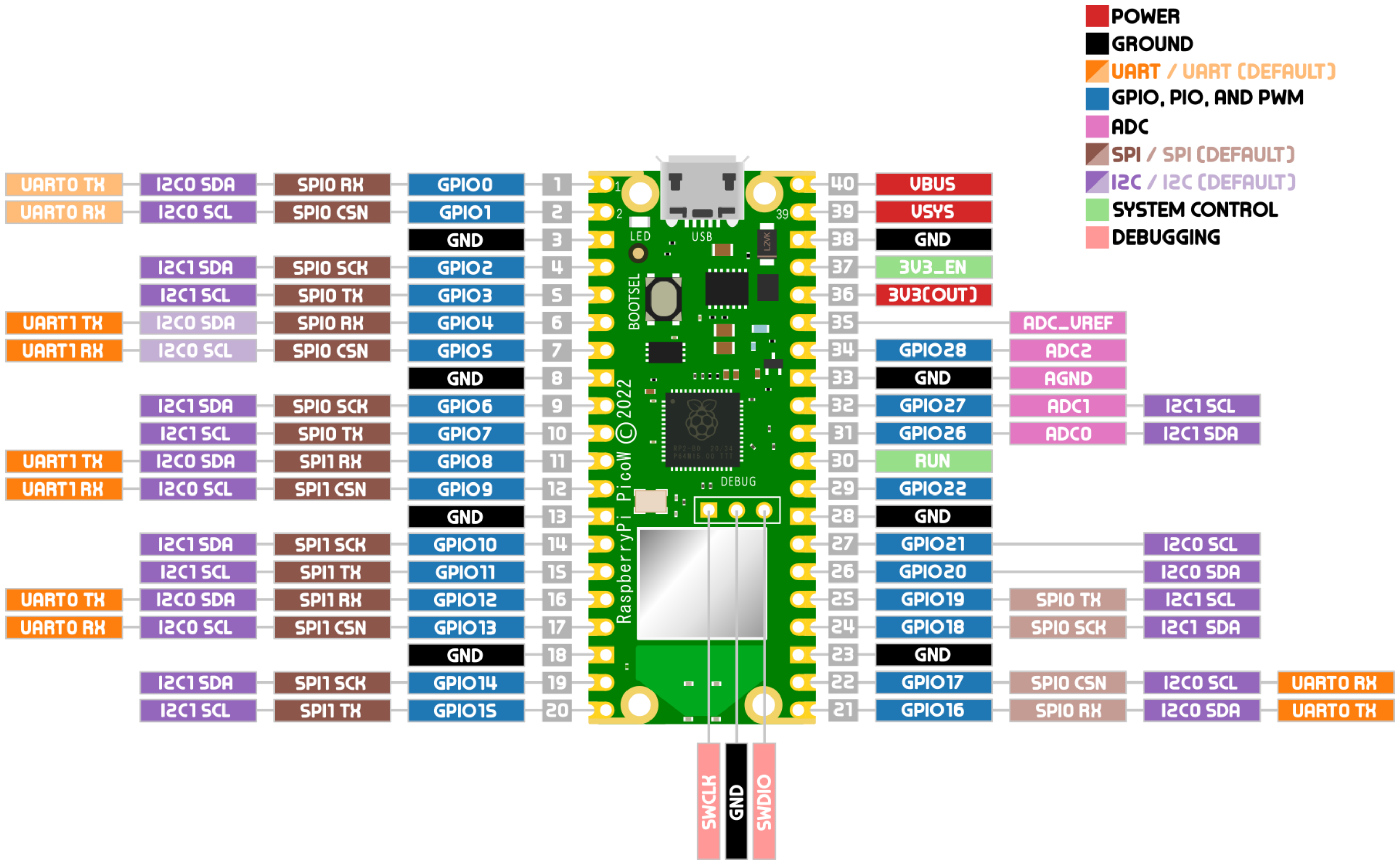


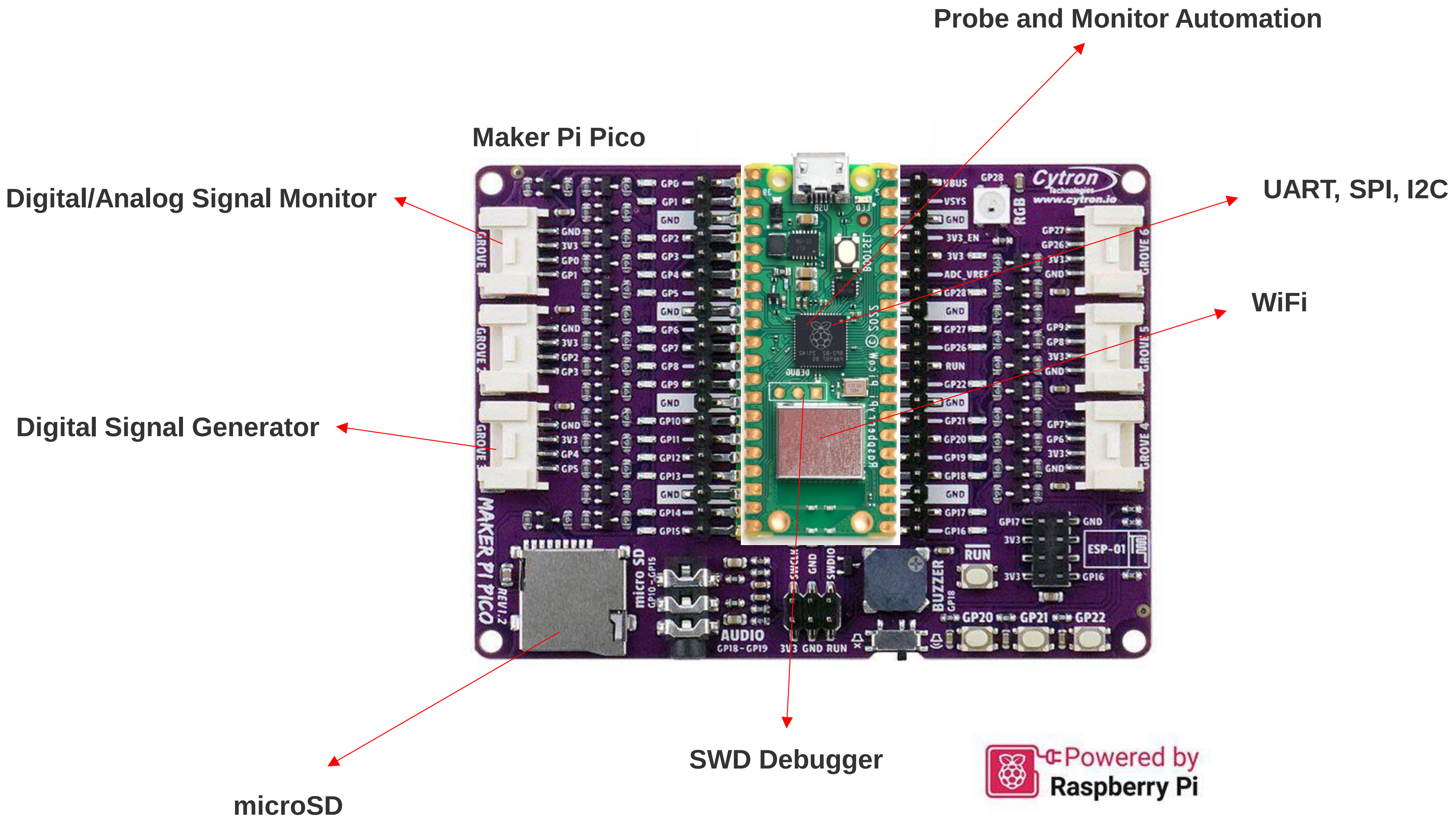
Pico Pirate Project Briefing

Reconfigurable Embedded Forensic Tool

Project Overview

- This project aims to design and develop an advanced embedded forensics tool based on the Raspberry Pi Pico, building on “Bus Pirate”. The primary motivation for this project is to enhance and extend the capabilities of the traditional Bus Pirate by addressing its limitations, such as the dependency on a PC for configuration and control.
- The Pico Pirate tool will offer improved functionality, including the ability to be configured and executed independently of a PC, making it more versatile and portable. Additionally, the tool will support remote control capabilities, allowing users to operate it wirelessly from a distance. It is particularly useful in fieldwork or scenarios where direct physical access to the tool is limited.
- By leveraging the powerful yet compact Raspberry Pi Pico platform, the Pico Pirate will provide comprehensive support for multiple communication protocols (e.g., UART, I2C, SPI) and enable forensic analysis of embedded systems with enhanced flexibility and user control. This project will focus on creating a user-friendly, standalone device that offers both local and remote operation, making it a valuable tool for professionals and hobbyists in embedded systems debugging and forensics.
- Team Composition: 5 buddies
- Each buddy leads on one of the key components while collaborating closely with team members on integration and testing.
    - Buddy 1: Data Capture & Logging
    - Buddy 2: Digital/Analog Signal Monitor and Analysis
    - Buddy 3: Digital Comms + Signal Generator + Automation
    - Buddy 4: SWD Debugger
    - Buddy 5: WiFi-based Remote Dashboard & Control


🟥 POWER
⬛ GROUND
🟧 UART / UART (DEFAULT)
🟦 GPIO, PIO, AND PWM
🟪 ADC
🟫 SPI / SPI (DEFAULT)
🟪 I2C / I2C (DEFAULT)
🟩 SYSTEM CONTROL
🟥 DEBUGGING
UART0 TX | I2C0 SDA | SPI0 RX | GPIO0 | 1
UART0 RX | I2C0 SCL | SPI0 CSN | GPIO1 | 2
GND | 3
I2C1 SDA | SPI0 SCK | GPIO2 | 4
I2C1 SCL | SPI0 TX | GPIO3 | 5
UART1 TX | I2C0 SDA | SPI0 RX | GPIO4 | 6
UART1 RX | I2C0 SCL | SPI0 CSN | GPIO5 | 7
GND | 8
I2C1 SDA | SPI0 SCK | GPIO6 | 9
I2C1 SCL | SPI0 TX | GPIO7 | 10
UART1 TX | I2C0 SDA | SPI1 RX | GPIO8 | 11
UART1 RX | I2C0 SCL | SPI1 CSN | GPIO9 | 12
GND | 13
I2C1 SDA | SPI1 SCK | GPIO10 | 14
I2C1 SCL | SPI1 TX | GPIO11 | 15
UART0 TX | I2C0 SDA | SPI1 RX | GPIO12 | 16
UART0 RX | I2C0 SCL | SPI1 CSN | GPIO13 | 17
GND | 18
I2C1 SDA | SPI1 SCK | GPIO14 | 19
I2C1 SCL | SPI1 TX | GPIO15 | 20
40 | VBUS
39 | VSYS
38 | GND
37 | 3V3_EN
36 | 3V3(OUT)
35 | ADC_VREF
34 | GPIO28 | ADC2
33 | GND | AGND
32 | GPIO27 | ADC1 | I2C1 SCL
31 | GPIO26 | ADC0 | I2C1 SDA
30 | RUN
29 | GPIO22
28 | GND
27 | GPIO21 | I2C0 SCL
26 | GPIO20 | I2C0 SDA
25 | GPIO19 | SPI0 TX | I2C1 SCL
24 | GPIO18 | SPI0 SCK | I2C1 SDA
23 | GND
22 | GPIO17 | SPI0 CSN | I2C0 SCL | UART0 RX
21 | GPIO16 | SPI0 RX | I2C0 SDA | UART0 TX
LED
USB
BOOTSEL
DEBUG
Raspberry Pi PicoW ©2022
SWCLK
GND
SWDIO


Probe and Monitor Automation
Maker Pi Pico
Digital/Analog Signal Monitor
UART, SPI, I2C
WiFi
Digital Signal Generator
microSD
SWD Debugger
Powered by Raspberry Pi

Data Capture and Logging

- The Pico Pirate’s data logging system functions like a mini-database, organising captured data in a structured, table-like format with precise timestamps and multi-dimensional categorisation. Users can perform custom queries, filter and sort data, and retrieve specific subsets efficiently. The system supports real-time querying and live data streaming. With features like a database-like user interface, the Pico Pirate enables in-depth forensic analysis and troubleshooting of embedded systems, providing users a powerful and versatile tool.
- Capture real-time data from various sources, i.e. communication protocols (e.g., UART, I2C, SPI), ensuring accuracy and integrity.

Digital/Analog Signal Monitor and Analysis

- The Signal Monitoring and Analysis is a versatile tool designed to monitor and capture both analog and digital signals from embedded systems. It provides comprehensive analysis and insight into the behavior of these signals, making it an invaluable resource for debugging and forensic analysis.
- **1. Analog Signal Monitoring:**
    - **Signal Digitization:** The tool converts analog signals into digital form, allowing for detailed analysis and interpretation. It captures the amplitude of the signal over time, creating a digital representation that can be further processed.
    - **Frequency Analysis (FFT):** The tool performs a Fast Fourier Transform (FFT) on the digitized signal, providing a detailed frequency spectrum analysis. This helps identify the dominant frequencies within the signal, useful for diagnosing issues related to noise, signal integrity, or interference.
    - **Signal Metrics:** Additional metrics such as the signal's root mean square (RMS) value, peak-to-peak voltage, and signal-to-noise ratio (SNR) are calculated to give a comprehensive understanding of the analog signal's characteristics.
- **2. Digital Signal Monitoring:**
    - **PWM Detection:** For digital signals, the tool can detect if the signal is a Pulse Width Modulated (PWM) signal. It calculates and displays the duty cycle (the percentage of one cycle in which the signal is active) and the frequency of the PWM signal. This is essential for analyzing motor control signals, LED dimming controls, and other applications where PWM is commonly used.
    - **Protocol Identification:** If the digital signal belongs to a communication protocol (e.g., UART, I2C, SPI), the tool automatically identifies the protocol in use. It decodes the data being transmitted and provides detailed information about the communication, including bit rate, packet structure, and any errors detected in the signal.

Digital Signal Generator

- **Adjustable signal parameters** such as frequency, amplitude, and timing to test and troubleshoot different scenarios.
- **Support for custom signal generation** scripts to emulate complex pulse sequences.

UART & I2C/SPI Communication Protocols

- The goal of this project is to develop and implement communication protocols (UART, I2C, and SPI) on the Raspberry Pi Pico, enabling it to interface with a variety of embedded devices.
- Additionally, the project includes the development of a baud rate detector for UART communication, allowing the system to automatically detect and adapt to different baud rates in real-time. Incorporate error detection and handling mechanisms, such as parity checking and frame error detection, to ensure reliable communication.

Probe and Monitor Automation

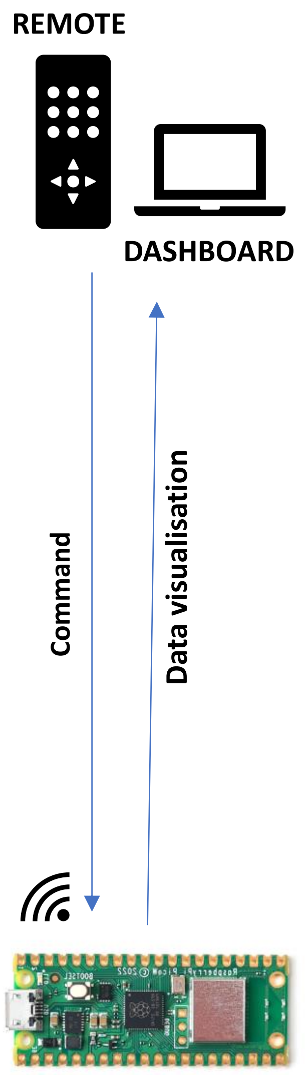
- This module automates the process of probing and monitoring signals within embedded systems.
- It systematically sends test signals via GPIOs, captures the corresponding responses through various interfaces like UART, SPI, and I2C, and analyses the results in real time.
- The system can detect specific patterns, measure signal characteristics (such as PWM duty cycle or frequency), and identify communication protocols.
- Designed for automated testing and debugging, this module streamlines the process of verifying signal integrity and system behaviour, making it a powerful tool for embedded system developers.

SWD Debugger

- The aim of this project is to develop a fully functional Serial Wire Debug (SWD) debugger implemented entirely on the Raspberry Pi Pico. This SWD debugger will be capable of interfacing with another Pico for debugging purposes. The debugger will accept commands either remotely or through simple, predefined commands, enabling developers to perform essential debugging tasks without the need for additional hardware or complex setups.
- **Low-Level SWD Handling:** Implement the SWD protocol on the Pico using its GPIO pins to directly interface with the target microcontroller's SWDIO and SWCLK lines. The implementation should handle the bit-level communication required by the SWD protocol, including JTAG-to-SWD switching, read/write transactions, and error handling.
- Simple Command Set: Define a set of simple commands that can be used to perform basic debugging tasks such as reading/writing memory, setting breakpoints, stepping through code, and halting or resuming the target microcontroller. Example commands might include READ_MEM, WRITE_REG, STEP, HALT, etc.

WiF-based Remote Dashboard and Control

- Enable WiFi connectivity to allow remote control and operation of the tool from a distance. Develop a remote interface (web-based) for configuring, controlling, and monitoring the tool wirelessly. Support remote data logging, signal capture, and signal generation, making the tool versatile for fieldwork or scenarios where direct physical access is limited.
- Dashboard Display: The tool provides a real-time display of the monitored signals, allowing users to observe changes instantly. The digital signals are represented with their decoded values, while analog signals are shown as waveforms with corresponding frequency analysis. The tool generates comprehensive reports summarising the captured data, including graphical representations of waveforms, frequency spectra, and protocol activity logs. These reports can be exported in various formats for documentation or further analysis.



Questions?