XFS4IoT SP-Dev Workgroup

1st April 2025

XFS4IoT SP-Dev Workgroup agenda

- Recap from previous meeting
- Framework roadmap
- Demo - XFS3 running along XFS4IoT
- What’s next?
- Next meeting

ATM Software

Recap from previous meeting

Recap from previous meeting

- Framework update
  — Version 3.0 published and available on GitHub
  — Reviewed main changes
  — Discussed breaking changes in spec version 2024-03

- Support for new forms in JSON

- Demo: working with forms

ATM Software

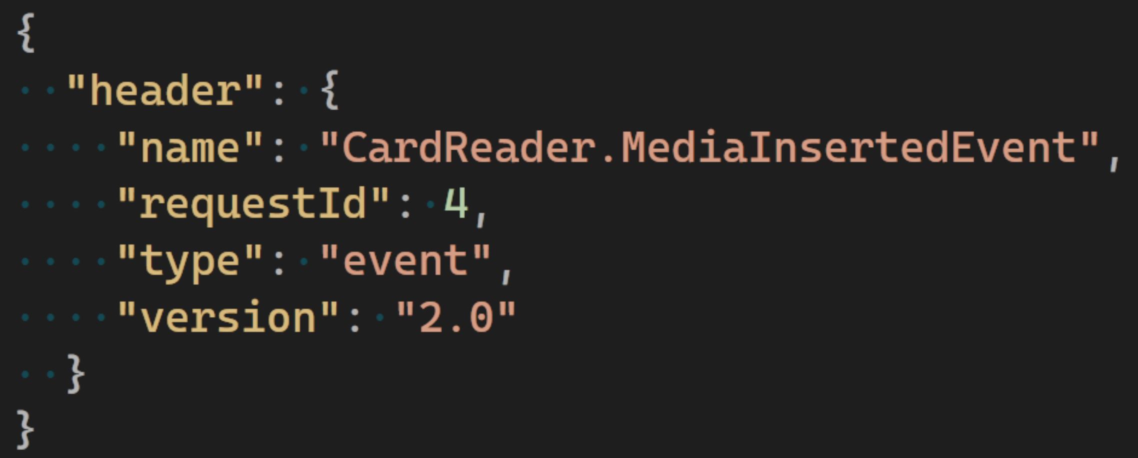
Framework Roadmap

Roadmap - Experimental ideas

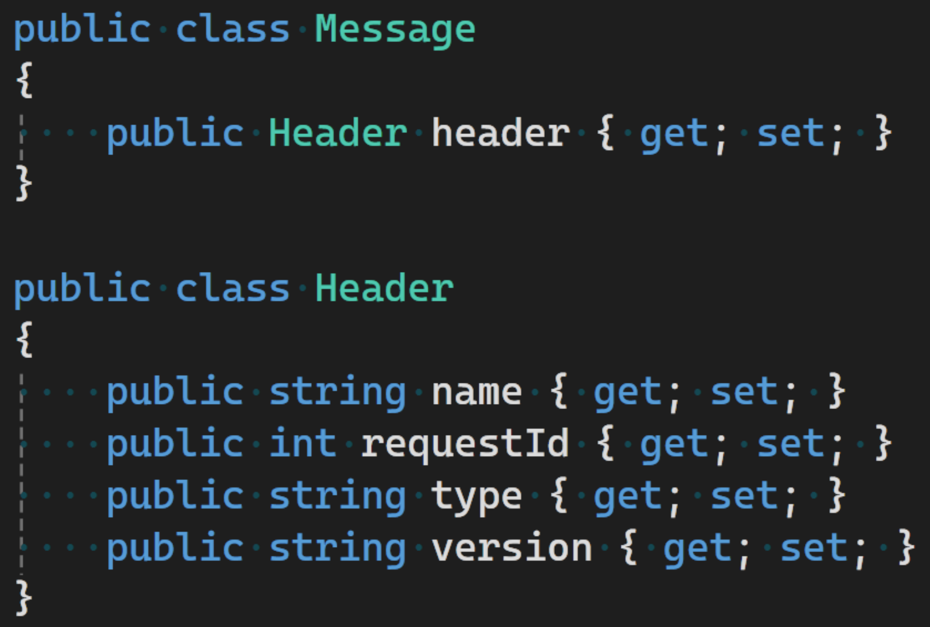
- Remove reflection – use code generators
- Use “Ahead of Time” (AoT) compilation
- Use small foot-print hardware…

Reflection - What is reflection?

- Dynamically create code at runtime
  - e.g. System.Text.Json is used to dynamically convert between JSON and C# objects using reflection



```json
{
  "header": {
    "name": "CardReader.MediaInsertedEvent",
    "requestId": 4,
    "type": "event",
    "version": "2.0"
  }
}
```



```csharp
public class Message
{
    public Header header { get; set; }
}

public class Header
{
    public string name { get; set; }
    public int requestId { get; set; }
    public string type { get; set; }
    public string version { get; set; }
}
```

Reflection

- System.Text.Json uses information about the classes (Message and Header) to convert to and from JSON
- Information is read and runtime using 'reflection' to dynamically work out how to map JSON to C# classes
- Can handle any class that maps to JSON

- Requires processing at runtime and information about the classes, even though everything is known at compile time

- A little bit slower, a little bit bigger

System.Text.Json and source generators

- Create the code at compile time, instead of runtime
- Possible with .NET 6 and later
- Faster startup. Lower memory usage. Better “assembly trimming”…
- System.Text.Json supports source generators
- We need source generators for CommandHandlers

Ahead of Time compilation

- .NET is a ‘managed’ language
  — C# is converted to a binary ‘intermediate language’ (IL) at compile time
- At runtime IL is converted to native code with “Just in Time” (JIT) compilation
- This includes standard code, like System, System.Text.Json etc.
- Slower startup, higher memory usage

Ahead of Time compilation

- Native AOT
  - — Convert C# directly to native Intel x64, ARM or other binaries
  - — No need for ‘JIT’ at runtime

- Assembly trimming
  - — Libraries like System and System.Text.Json are also compiled with AOT
  - — Libraries and even parts of libraries that aren’t used are excluded

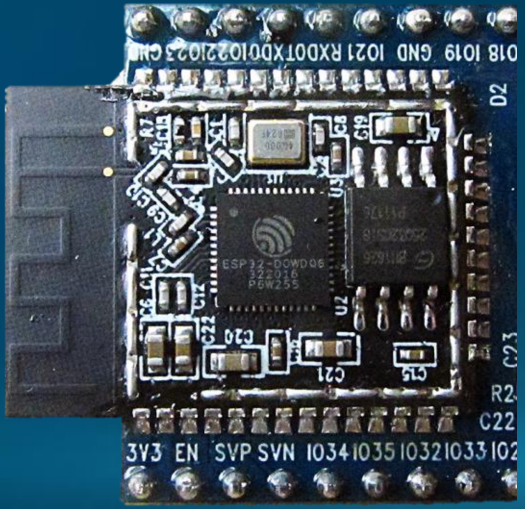
- Self-contained
  - — No need for external dependencies. Small and fast.

Roadmap – Embedded devices

- Remove reflection – Less memory, better “Assembly trimming”
- Native AOT – Less memory, self-contained
- Assembly trimming – Less memory, smaller storage

- How small can we go?

- .NET nanoFramework – “MicroController Units” (MCU)



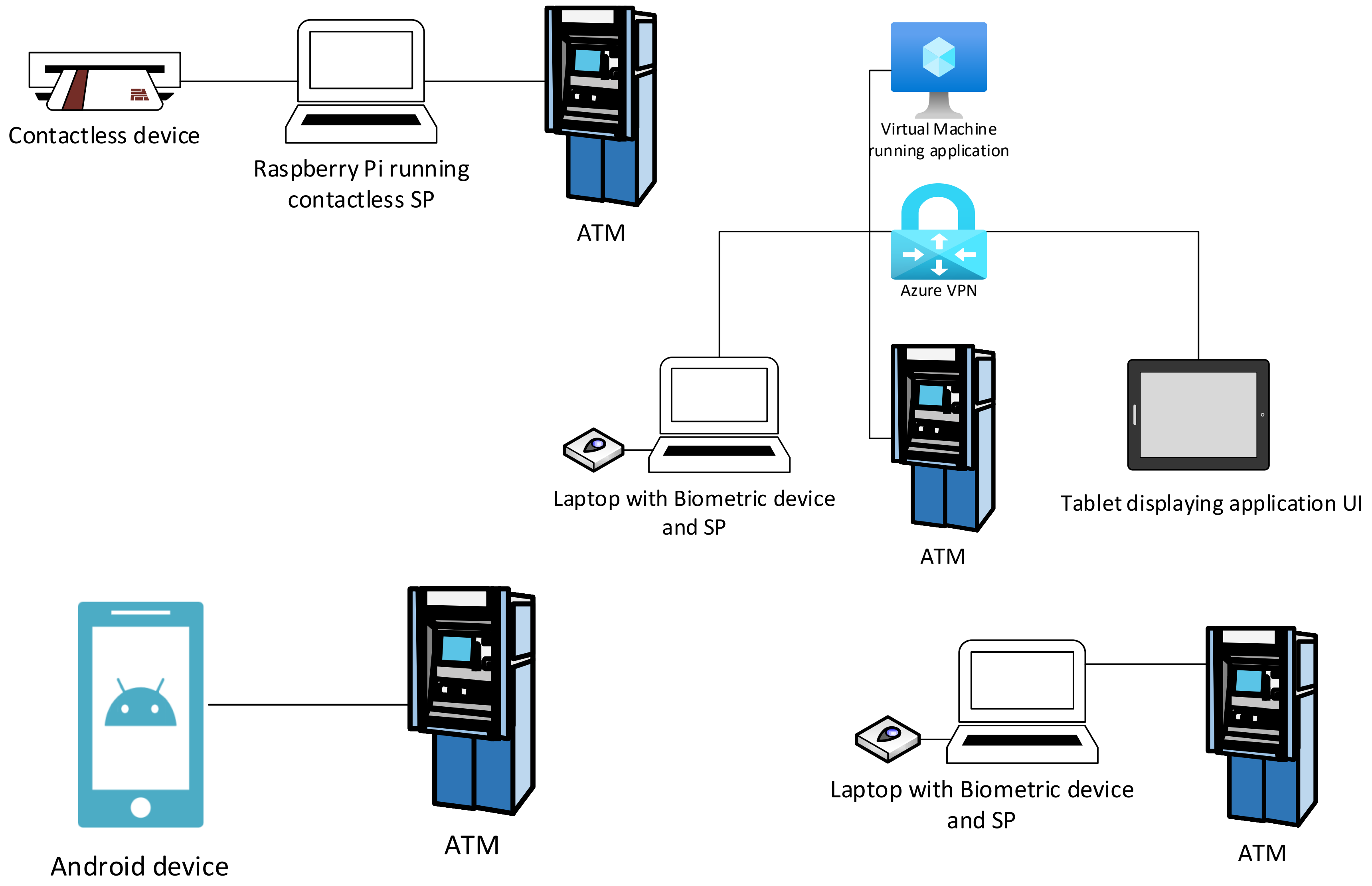
ATM Software

Demo - XFS3 + XFS4IoT

Possibilities with XFS4IoT and existing hardware

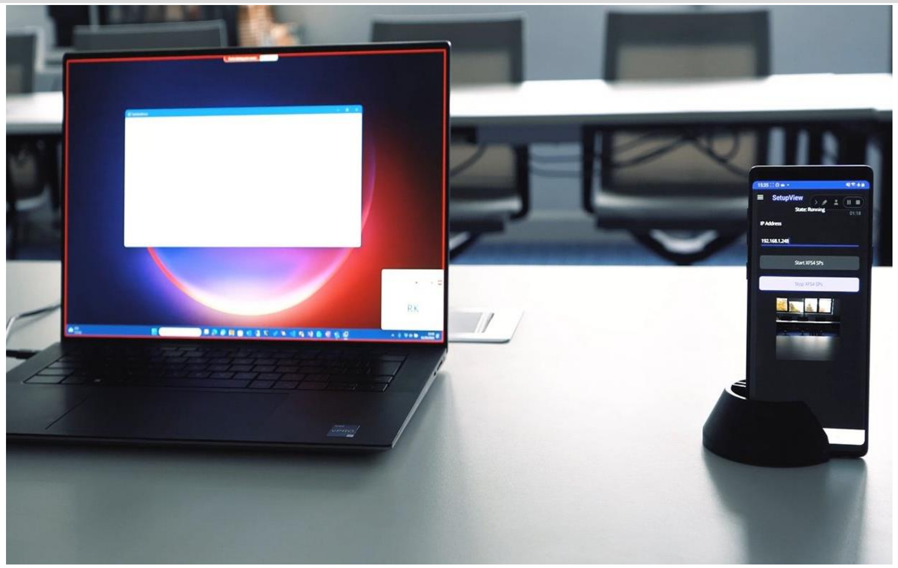
- Adding new devices to existing hardware

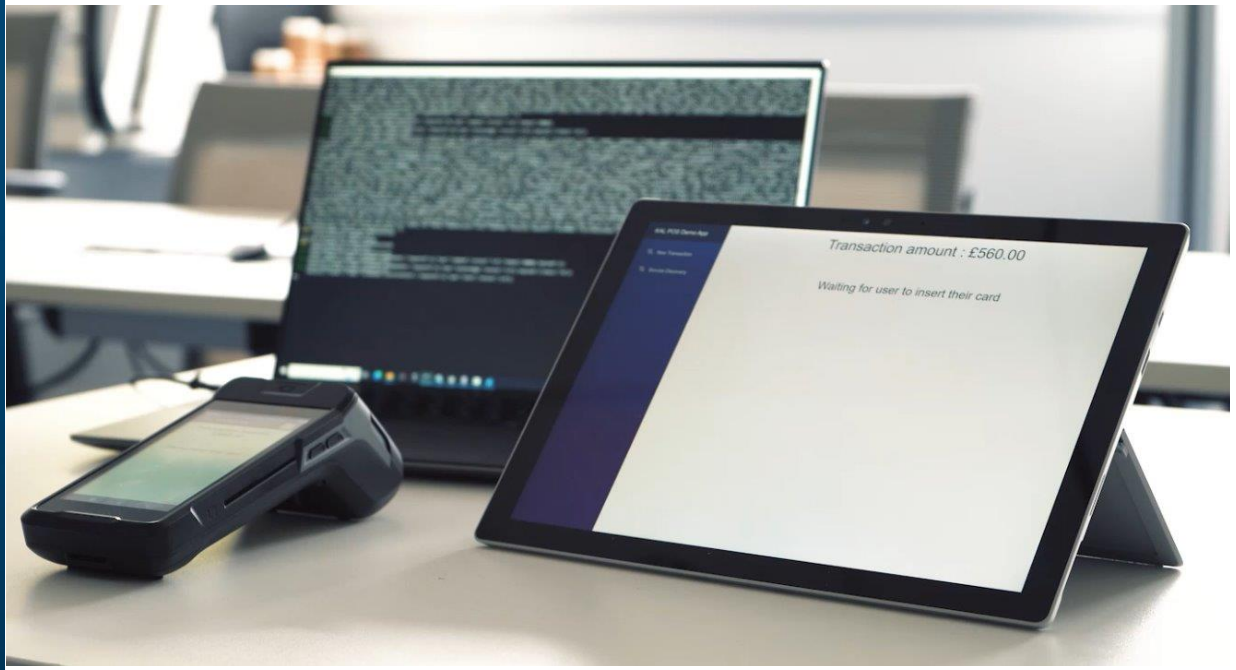
- Moving applications to the cloud

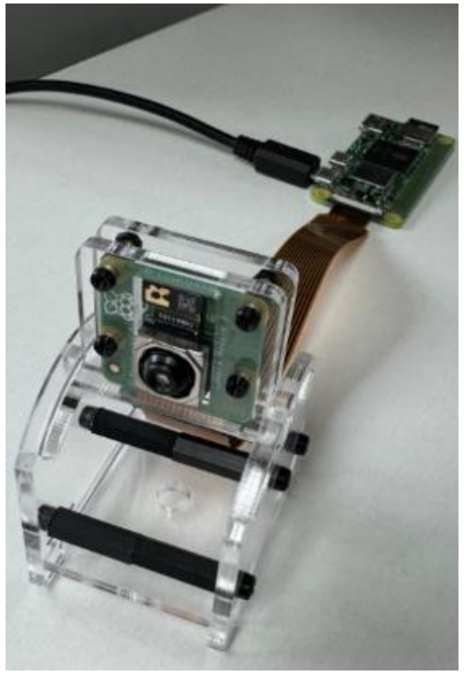

Contactless device
Raspberry Pi running contactless SP
ATM
Virtual Machine running application
Azure VPN
Laptop with Biometric device and SP
ATM
Tablet displaying application UI
Android device
ATM
Laptop with Biometric device and SP
ATM

Examples from previous demos

- Biometrics device
- Raspberry Pi and Android camera devices
- POS device



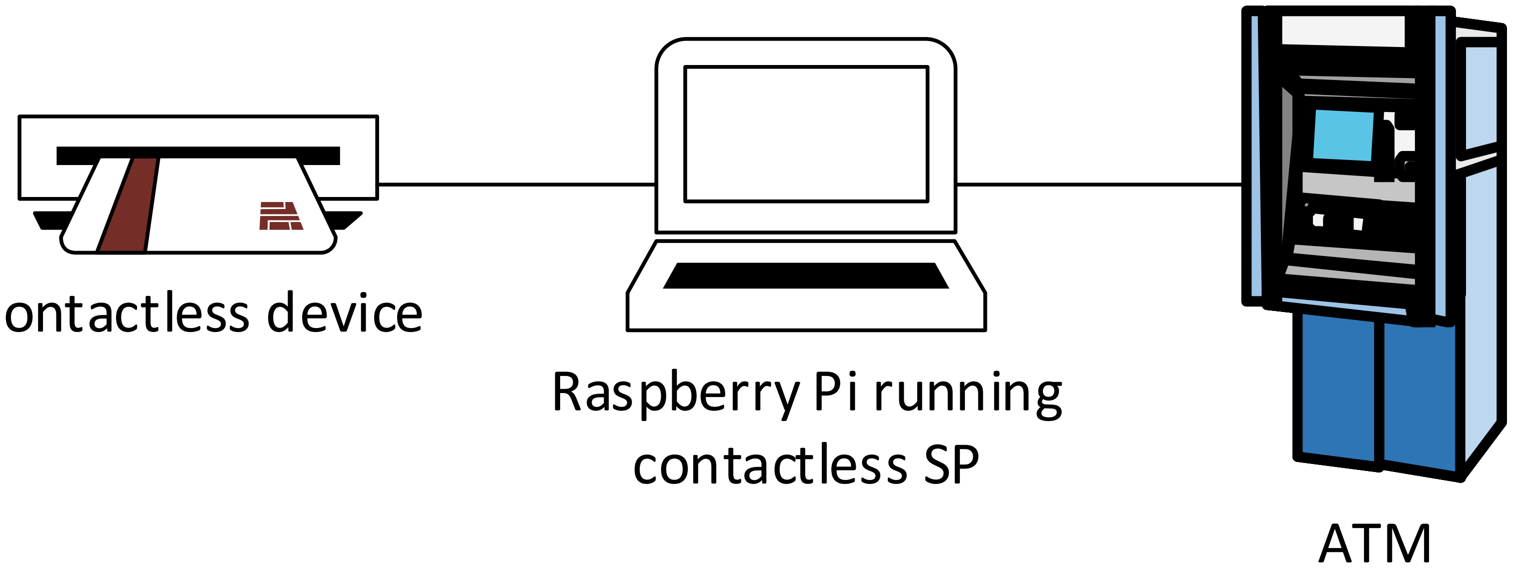

Transaction amount : £560.00
Waiting for user to insert their card



The demo

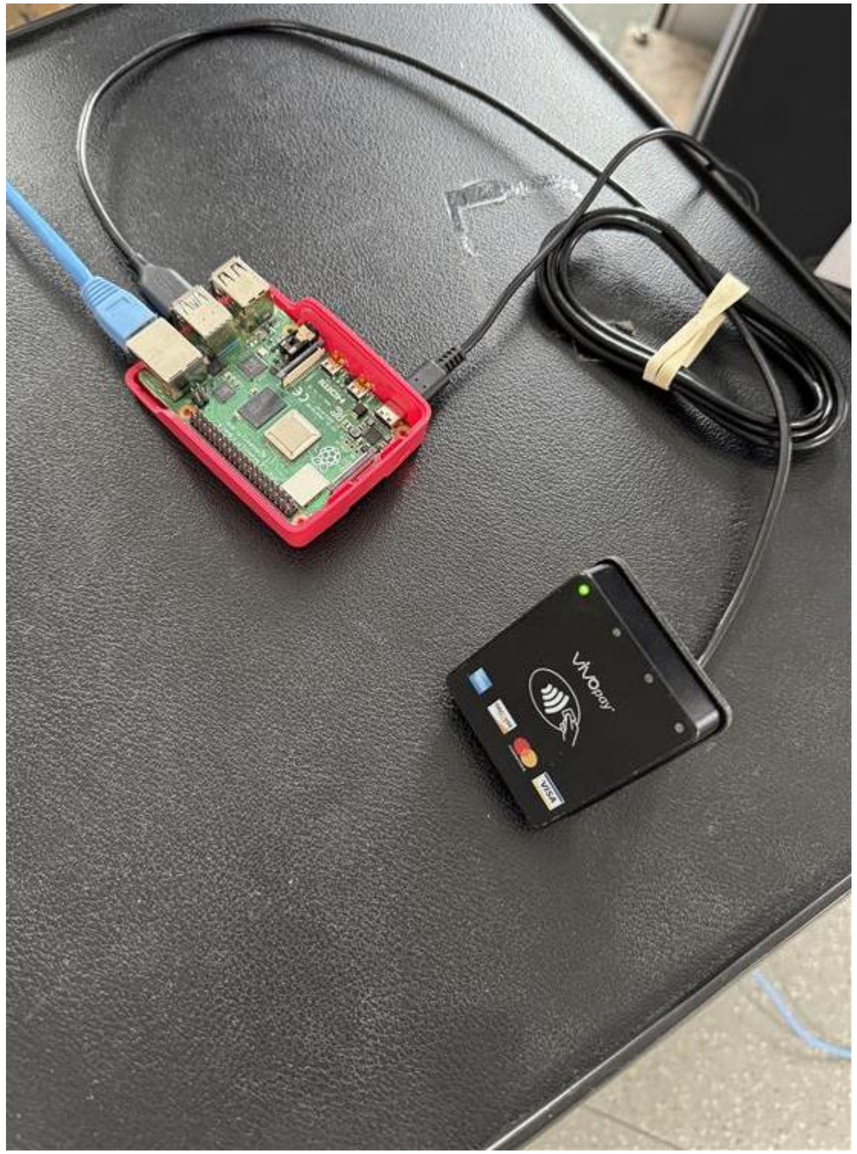
- Application running on ATM

- XFS4IoT Service Provider connection over wired network



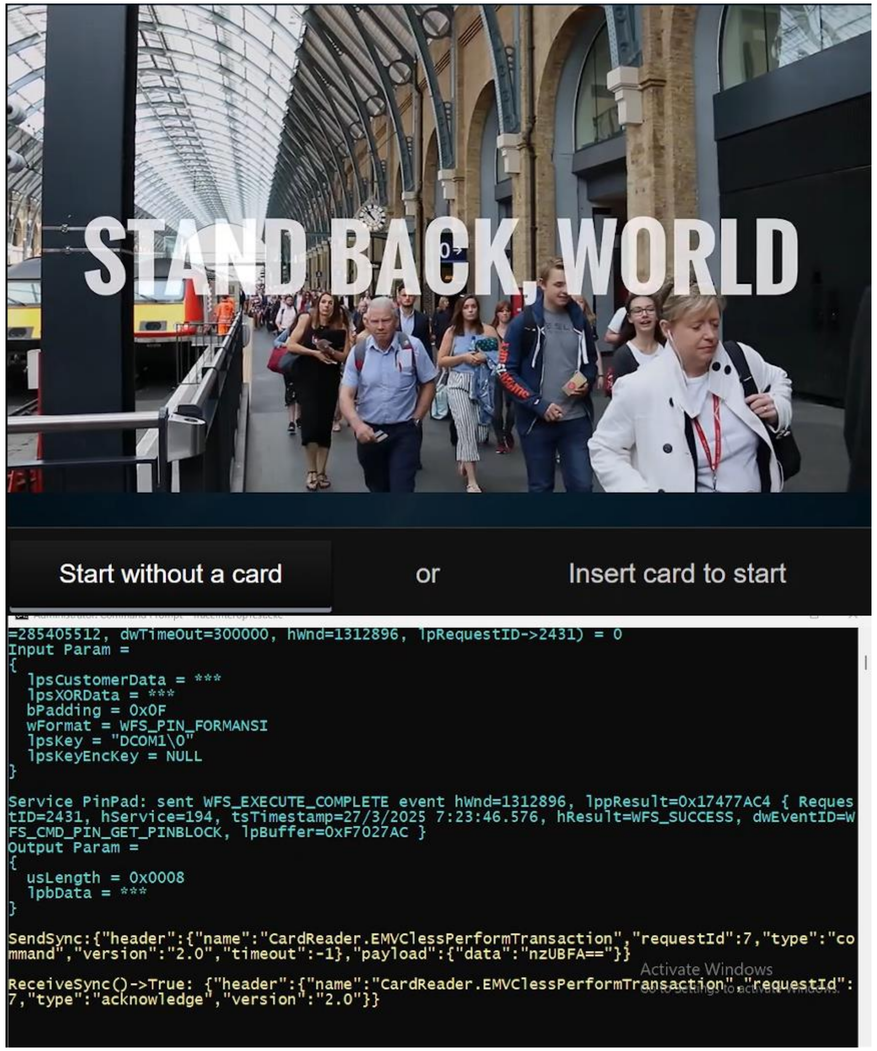
Contactless SP

- Raspberry Pi 4
- Contactless card reader
- XFS4 CardReader Service Provider



Demo

- XFS3 PinPad Service Provider
- Window with live XFS 3 and XFS4IoT messages



```
=285405512, dwTimeOut=300000, hWnd=1312896, lpRequestID->2431) = 0
Input Param =
{
  lpsCustomerData = ***
  lpsXORData = ***
  bPadding = 0x0F
  wFormat = WFS_PIN_FORMANSI
  lpsKey = "DCOM1\0"
  lpsKeyEncKey = NULL
}

Service PinPad: sent WFS_EXECUTE_COMPLETE event hWnd=1312896, lppResult=0x17477AC4 { Reques
tID=2431, hService=194, tsTimestamp=27/3/2025 7:23:46.576, hResult=WFS_SUCCESS, dwEventID=W
FS_CMD_PIN_GET_PINBLOCK, lpBuffer=0xF7027AC }
Output Param =
{
  usLength = 0x0008
  lpbData = ***
}

SendSync:{"header":{"name":"CardReader.EMVClessPerformTransaction","requestId":7,"type":"co
mmand","version":"2.0","timeout":-1},"payload":{"data":"nzUBFA=="}}

ReceiveSync()->True: {"header":{"name":"CardReader.EMVClessPerformTransaction","requestId":
7,"type":"acknowledge","version":"2.0"}}
```

ATM Software

Demo video

ATM Software

What's next?

What’s next?

- Framework updates and roadmap
- IBNS
- Guest speakers
- More demos (biometrics and more)

Next call

Zoom

- First Tuesday of each month at 1300 UK time for 30 mins

Next call: 6th May 2025
1300 UK, 0800 US EDT, 2100 Tokyo time

Calls are 30 mins long

We will continue to use Zoom
(Interpretation in Japanese, Chinese and Spanish is available using Zoom's interpretation feature)