

Developing IoT Applications Using MATLAB

Dr Samuel Bailey, Skyrad



Background

- Skyrad are small engineering company – focus on R&D and systems engineering
- Initial product was Skyrad Solar Space Heating – developed from a Simulink Dynamic model of home heating performance.
- Since invented several IoT devices using MATLAB in various configurations



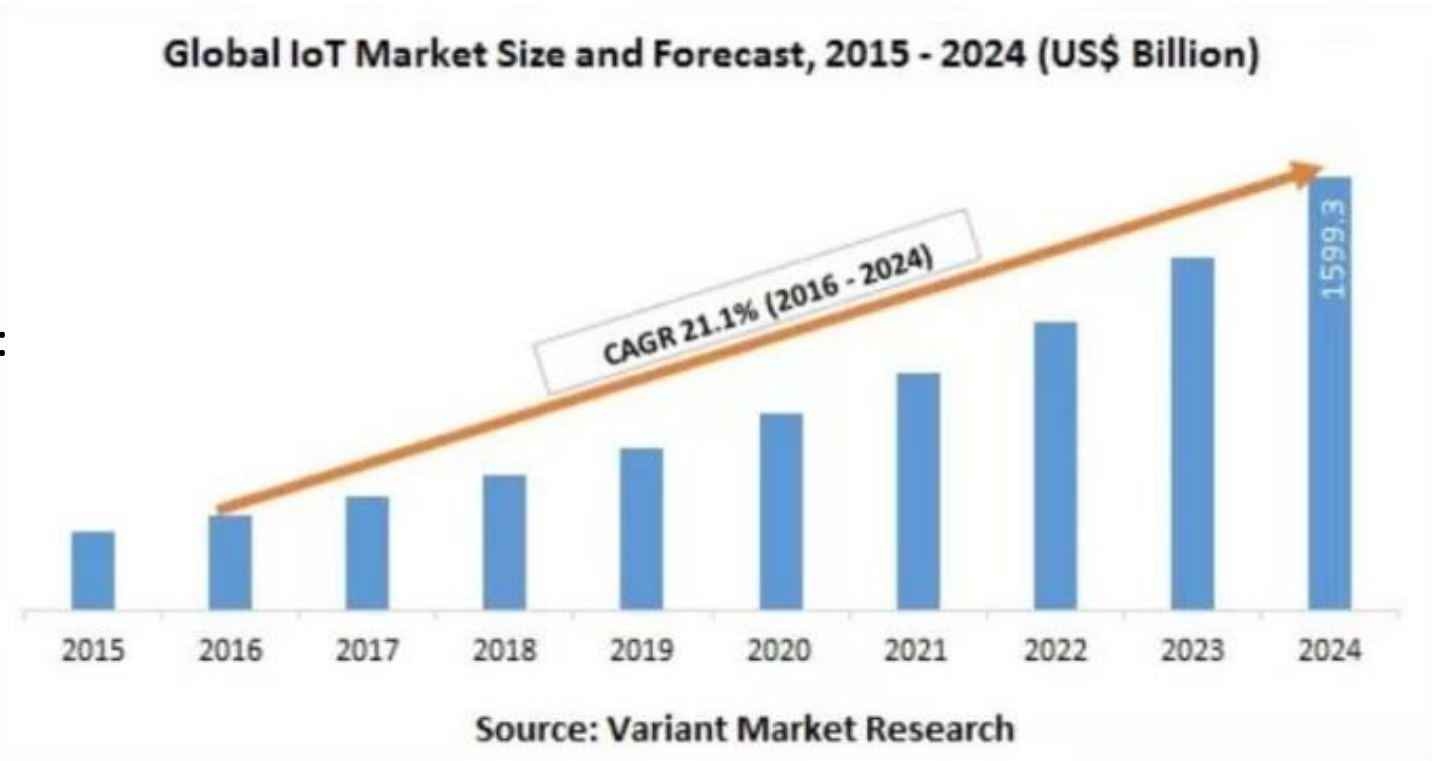
HeatDoctor

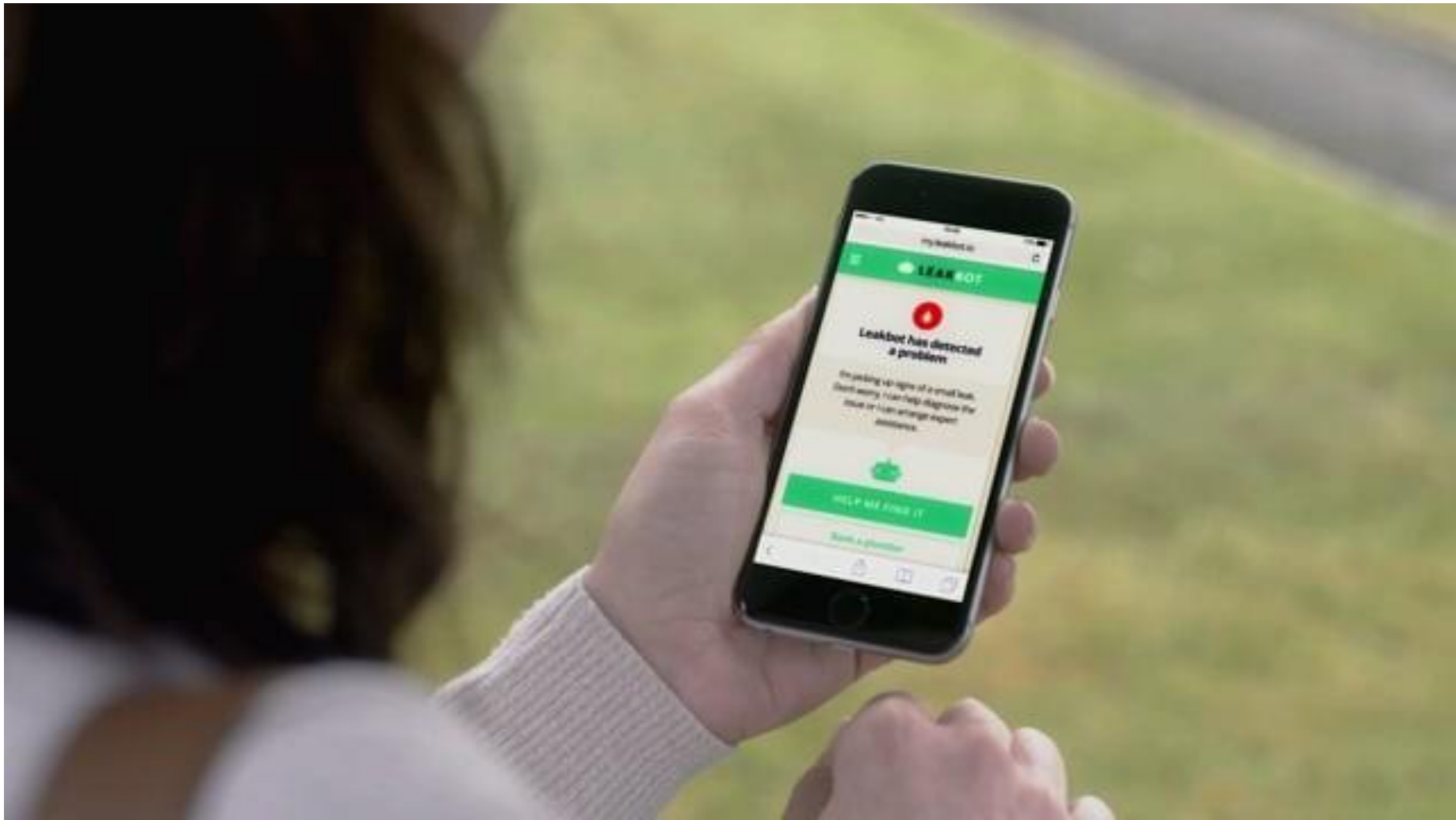
pipa

SKYRAD

IoT Industry

- Massive growth predicted (again)
- Main players are the big cloud computing platforms: AWS, Microsoft Azure, Google Cloud
- Limited overlap with the embedded world. Standard architecture assumes:
 - Dumb edge sensor running Linux
 - Continuously available internet connection
 - Cloud based analytics
- We have used the cloud platforms for infrastructure, but not their architecture.
- Instead taken several different approaches using the MathWorks tool chain



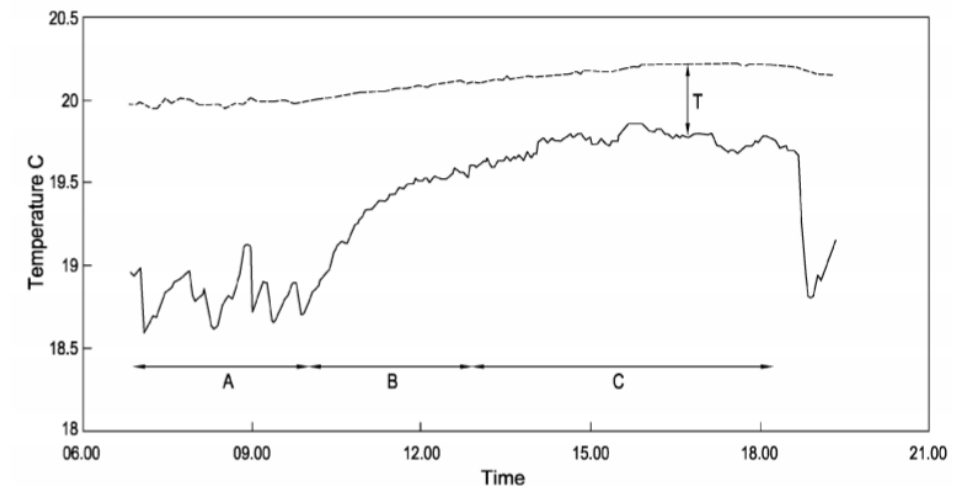
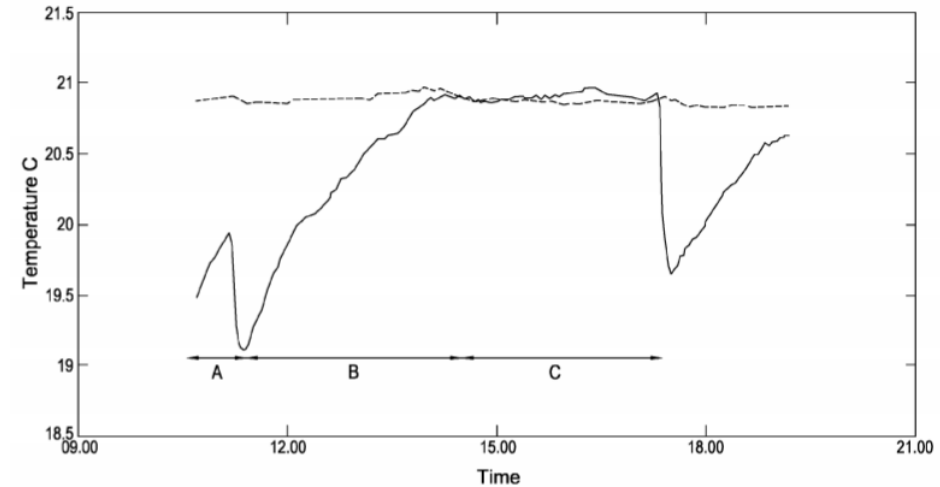




LEAKBOT®

How LeakBot Works

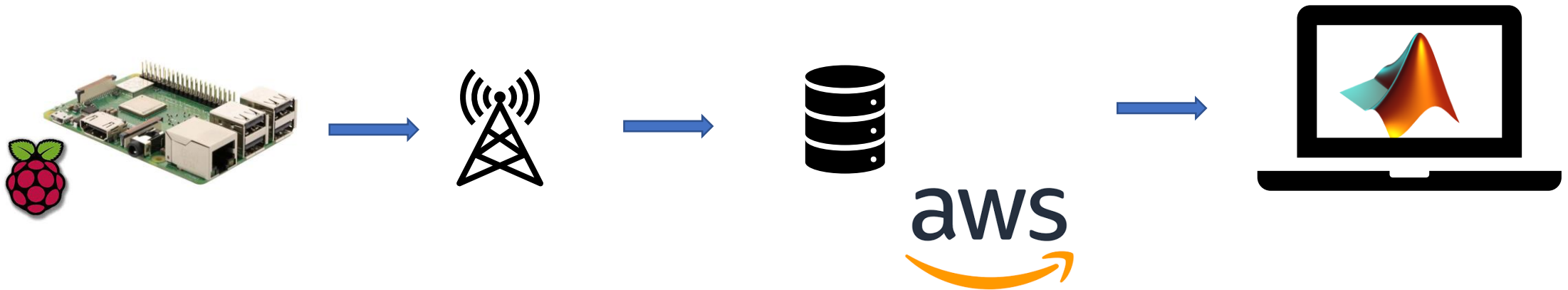
- Water (usually) enters your house slightly colder than ambient temperature
- Temperature difference can be exploited to estimate flow rate in the pipe.
- If there is constant flow for a period of time, chances are it is a leak (though can be a dripping tap or a hose pipe)
- Detects 98% of leaks over 5ml/min.



SKYRAD

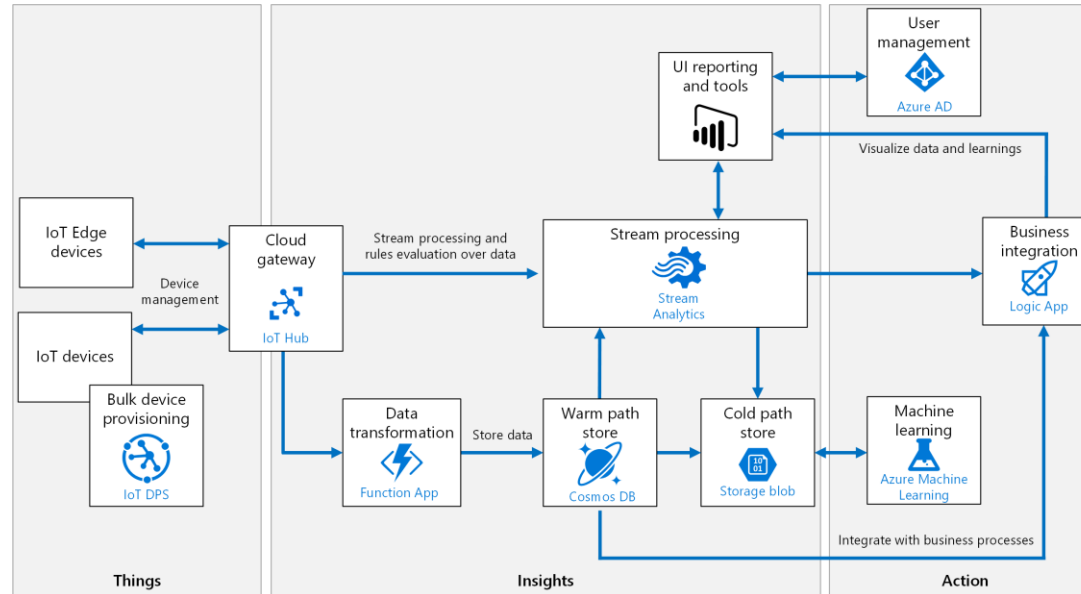
Evolving LeakBot Architecture

- Proof of Concept – started as an IoT Device so we can develop algorithms remotely



- Proved concept works, developed algorithms and patented. Client gave go ahead to develop.

This Follows the Standard IoT Model



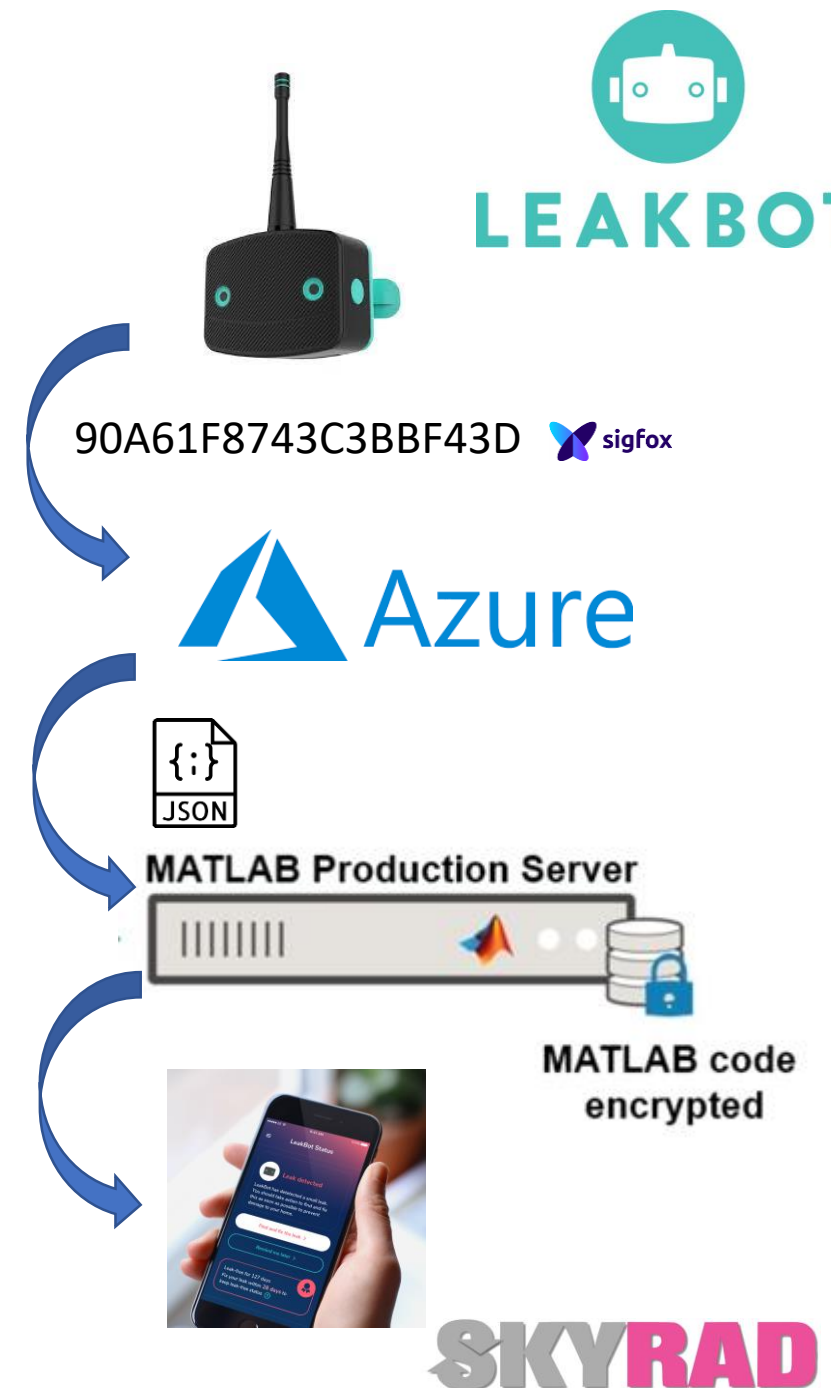
Which assumes the IoT device is running an OS, and continuously powered. But we need to run on batteries, and in 2014 we weren't sure people really needed connectivity

Evolving LeakBot Architecture

- V1 – standalone leak detector.
- No connectivity, 5 year battery life, beeps when a leak is detected
- MATLAB algorithm from PoC hand coded in C to run on embedded hardware.
- Embedded device logs all its data. Plumbers have a tool to download the data and send it back, so we can adapt future code to remove classification errors.

Evolving LeakBot Architecture

- V2 – clients want data, so we added Sigfox connectivity
 - Very low bandwidth (12 bytes per day).
- Device pre-processes timeseries data then sends compressed data via Sigfox to cloud
- Cannot send much data, but want to remotely tune algorithm and optimise sensitivity
 - Initial cloud based rules run in C
 - More sophisticated algorithms developed in MATLAB, then run on MPS
 - Empirical and machine learning algorithms compared on cloud
- LeakBot app on phone

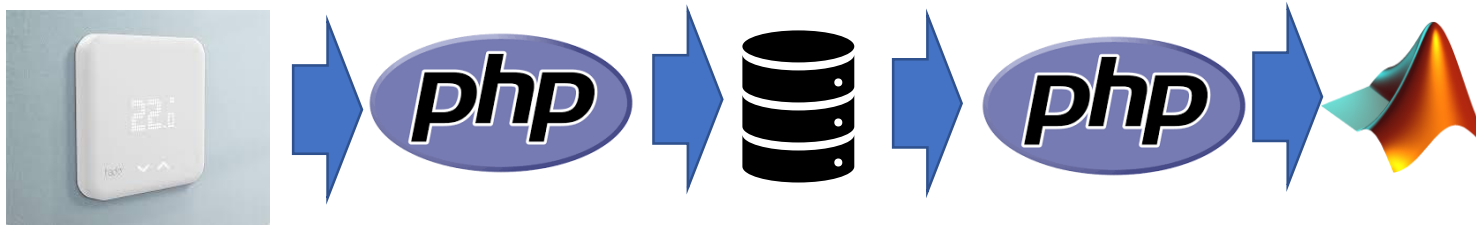


Evolving LeakBot Architecture

- V3 - WiFi connected LeakBot (Sigfox rollout stalled in UK)
- WiFi is high bandwidth but high power consumption
 - Can only be on for a few seconds a day
- Fast Leaks (burst pipes) detected locally.
- Other data buffered on the device and sent to cloud MPS every 24 hours
- MPS runs the existing proven, tested algorithm, so we only need minimal revalidation
 - Further enhancements tested against device logs, then copied to MPS to go live with minimal testing.

HeatDoctor

- Detects faults in you Central Heating by monitoring data from Smart thermostat (e.g. Nest, Tado)



- Initial architecture used php (on Apache web server) which called MATLAB algorithm on MPS

Scaling Up

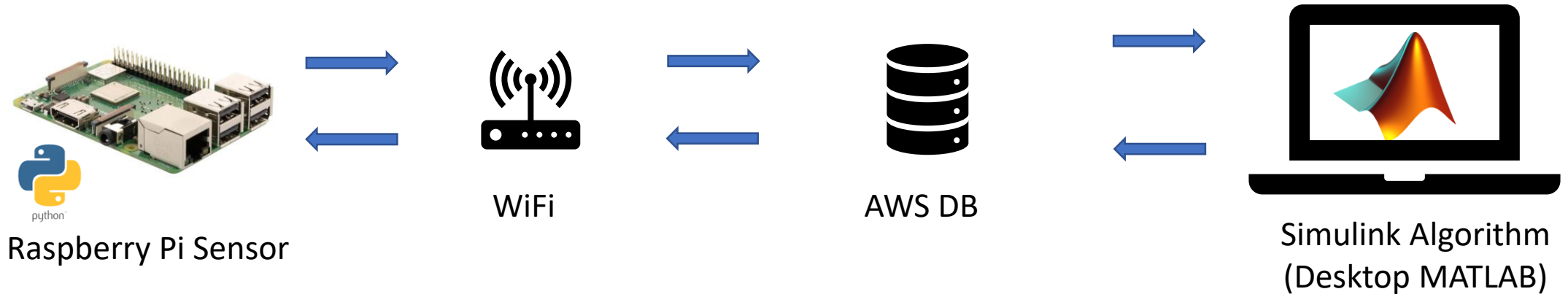
- MATLAB provides the same benefits for algorithm prototyping and development
- For this project, benefits of AWS Lambda outweigh time to recode algorithm for deployment stage
- In future we would like to see:
 - Hosted MPS for easier trials and PoCs
 - Lambda function equivalent on the cloud – MathWorks ‘Function as a Service’ where we don’t need to care about scaling, servers or RDPing in

Project Pippa – In home target tracking with thermal imaging sensor

- Based on lessons from LeakBot, how would we develop on a clean sheet of paper?
- Still need to run on batteries
- Still need connectivity
- Want to retain process of develop algorithms offline environment, then deploy to production with minimum risk and retesting.
- Device needs to respond quicker than LeakBot


Initial PoC – same architecture

- Prototype Simulink algorithm in the cloud using a dumb sensor



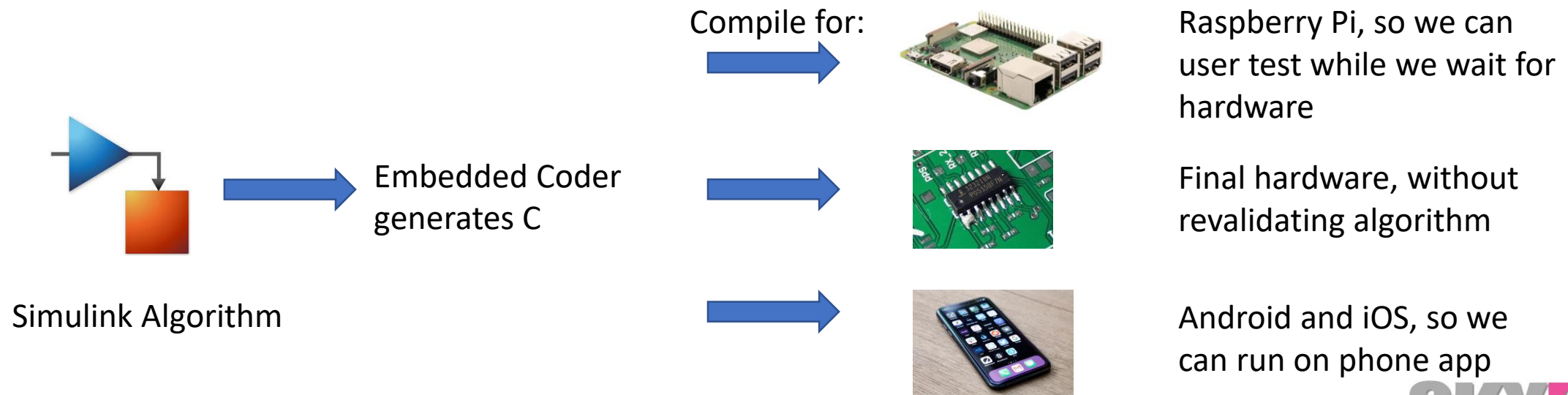
- Ran in 10 or so homes for 12 months gathering data.
- MATLAB writes results of analysis to AWS table
- Pi reads results and gives feedback to user.

Decisions

- Want device to still work for consumer even if not connected
- Want to minimise chance of not being connected even if user is not tech savvy
 - Cellular connectivity. £££ and  if using data, but cheap and low power on SMS.
- So need to run algorithm on embedded device.

Current (Final?) Architecture

- Simulink model refined on logged data.
- Code runs locally on embedded device. Outputs sent via SMS.
- Embedded coder used to generate C. Interface designed and tested against backend server.
- C code then compiled onto RPi so users have genuine experience of the device while hardware is developed.
 - Code optimised for memory use and compiled for target hardware.
 - Same code also compiled to form the intelligence within the associated app and run on Android and iOS.



Conclusions

Haven't hit volume yet, however:

- MATLAB/Simulink -> Embedded Coder -> Deploy to any type of firmware (Android/iOS, Raspberry Pi and final low power embedded device) was an enormous productivity boost compared to other architectures.
- Confident that algorithm behaves the same when run on historic data on desktop as on embedded device in real time.
- Can run compiled C on low cost battery powered devices without the overhead of an OS and without keeping a WiFi link open.
- **MATLAB ought to become a go-to solution for IoT development.**