Enabling CI/CD (DevOps) workflow for Hardware Production

Validate

Chandler Samuel Keep

A Dissertation presented for the degree of MSci
Computer Science (Hons)

In collaboration with

School of Computing and Communications
Lancaster University
Supervised by Prof. Joe Finney

June 2021

Abstract

This Project outlines the work on 'Validate', a software solution to hardware validation testing and electrical diagnostics for JACDAC hardware modules. It forms practical research to demonstrate the impact of enabling CI/CD (DevOps) in hardware production through the use of a standardised device serial protocol, JACDAC. This work is inspired and takes place as part of the 'Democratizing Hardware' [1] ***initiative at Microsoft Research. Our belief is that the work will encourage growth and innovation in the long-tail hardware market, as well as make hardware manufacturing a more accessible process to the masses. We explore the challenges currently facing hardware production and validation testing, how this affects accessibility, why such a ci/cd solution will encourage growth and explore implementing such a solution including its challenges.***

I. Introduction

This project takes place in collaboration with Microsoft Research (MSR) within their 'Democratizing Hardware' Initiative. The 'Democratizing Hardware' initiative focuses on the 2 phases of hardware development (see fig 1.), encompassing the goals of making easier, more accessible, and less costly the ideation & production phases of new hardware. The first phase of hardware development, ideation, can already be considered democratized [2]. Supported by a large research community, interactive hardware eco systems have been able to thrive such as Arduino, BBC Micro: Bit and Adafruit, making prototyping and designing new interactive devices more accessible than ever. Advancements have also been made by MSR to enable this, with .NET Gadgeteer previously, and now JACDAC [3], a plug and play hardware/software stack for embedded systems to support rapid prototyping of new devices.

Whereas the goal of democratizing the 2nd phase of hardware development however, production, is still an open research problem, and an area that is to be partly addressed by the proposed project. The focus of the 2nd phase is to be able to turn from a designed prototype to hundreds or thousands of manufacturable copies, either for evaluation (such as for research) purposes, or as a low-volume product. This process is often very challenging and ensuring reliable production as manufacturing scale grows becomes increasingly complex. Complexity is added through non-recurring engineering (NRE) activities, involving processes such as design verification and testing to ensure design replication consistency; altogether known as the 'replication challenge' [4].

The vision is therefore to empower long-tail hardware innovation, meaning encouraging development and making cost productive the development of a large number of niche but viable products. One way I propose to enable this, is through the development of a software CI/CD solution for hardware, to tackle one of the many issues of NRE activities, design verification and testing.

The aim of this project is to tackle this in the scope of MSRs upcoming embedded device ecosystem Jacdac. My goal is to design a universal Jacdac testing suite using software, allow CI/CD testing of JACDAC enabled devices. The testing suite will enable users to validate their modules conform to Jacdac standards, implement the protocol correctly and operate in non-electrical function specifications. Non-functional specifications include the electrical characteristics of the device under test (DUT), this includes whether it is pulling safe current under load etc. to avoid damage to a module. This is important to ensure there is no

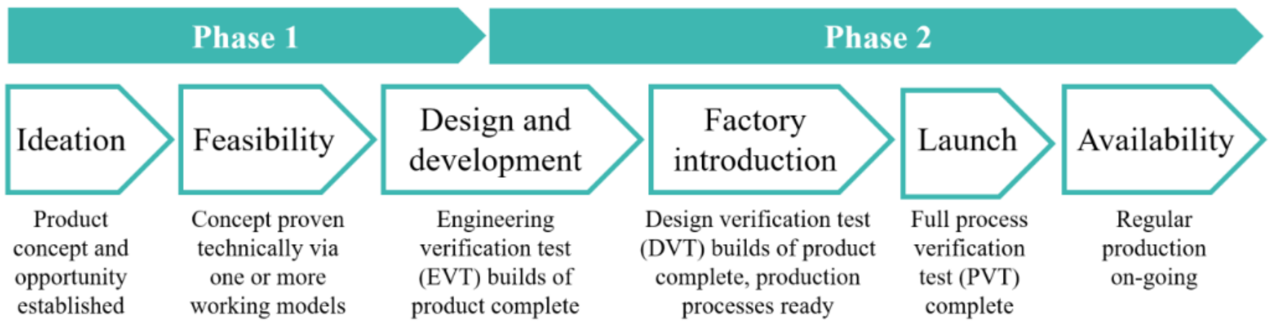

Phase 1
Phase 2
Ideation
Feasibility
Design and development
Factory introduction
Launch
Availability
Product concept and opportunity established
Concept proven technically via one or more working models
Engineering verification test (EVT) builds of product complete
Design verification test (DVT) builds of product complete, production processes ready
Full process verification test (PVT) complete
Regular production on-going

Figure 1: The two main phases of hardware development encompass six main activities. The transition from Phase 1 to Phase 2 typically happens when a handful of functional working prototypes have been made.

electrical faults with the board as well as bug in the firmware causing excessive current draw that could damage the device. The testing suite therefore will validate the component functionality, as well as on-board device electrical diagnostics, which could also be used for example to demonstrate accordance with regulatory requirements.

II. Problem definition

The major barrier for the democratization of the hardware production phase is the up-front cost and expertise required in non-recurring engineering activities. This process involves custom test jigs to be designed and built to validate each replica, as well as specialised labs for regulatory tests, adding significant cost for expertise and equipment requirements. The time taken to undergo these activities is usually longer than the manufacturing and design process of a prototype itself! Alongside these processes, the following NRE activities also have to be considered, including finding reliable suppliers for all components and materials, accommodating component tolerance, designing and building the necessary tooling, building an efficient and reliable manufacturing process, controlling and accommodating manufacturing variability, selecting and managing manufacturing partners, instigating and maintaining manufacturing quality control, and adapting to changes in pricing and availability of components and services [4]. All of these activities are important to address the replication challenge of variability in manufacturing, but all have huge monetary costs.

Currently in phase 2 hardware developers to accommodate validation testing must go through a DVT process (& design for testing) so that production is viable during the manufacture DVT step. DVT involves having to develop a test jig for the manufacturer, in order to run functional tests to validate each device after production. Alongside this, environmental testing and regulatory pre-compliance testing must be carried out, and any non-compliances found with the device will mean lengthy iterations & design stages having to be re-carried out. During the DVT stage (see fig 1) the manufacturer will use the designed test procedures from the developer and their own designed test jig to validate each unit from a production line to certify yields. Depending on the results, further iteration in product development or refinement of testing may be needed, before finally being able to go to the final step of production involving regulatory compliance certification.

Therefore, before development moves from prototype to the manufacturing phase it should be considered important to have designed thorough and extensive test requirements to avoid lengthy delays and cost in production. As studied by R.Khurana et al. [5] it is often considered challenging to design for testing & rather inaccessible without the technical expertise of the manufacturing process, which most small companies and individual developers won't have.

Due to the lack of coordination & inaccessibility between the design and production phases, there is no industry standard approaches [5] when it comes to the development of test jig hardware, or software functional test design for manufacturers to be carried out by developers. The process is only realised when hardware developers move to the production phase and start communicating specifications & building requirements with the manufacturers. This is a huge issue because manufacturers will all have different requirements, and test design will unlikely be consistent across manufacturers, as manufacturers themselves often build the test jig. This will also mean having to ultimately redesign to accommodate testing requirements set by the manufacturer at this stage-incurring again more costs and time delays that most developers simply cannot afford.

It is therefore important we hand back control of testing to the developers through a ci/cd hardware production process standard, that can be carried out & designed in phase 1 alongside initial prototype development. This will ensure that developers are able to design the prototype from the start with full test coverage in mind and avoid extra costs and time delays when they move to the manufacturing process. All of this will make the manufacturing process easier and more dynamic and will also ultimately enable freedom in choice to switch/choose manufacturers. Therefore, by providing a fully working testing process in advance independent from manufacturer influence, it also enables dynamic supply chains for developer's by being able to hop from manufacturers freely.

Therefore, reviewing the past problems the proposed ci/cd solution is required to improve accessibility and enable innovation & growth in the long-tail hardware space.

III. Background

Our focus for the 2 phases of hardware production (see fig 1) is MCU (Micro-controller) class devices that typically combines a MCU with 10's or hundreds of other electronic components via on or more PCB's. During step 2 [5] one or more functional prototypes validate feasibility of the product. The outcome of phase 1 EVT test is carried out to develop a "looks like works like" prototype to mark end phase 1 to phase 2. During phase 2 the functional prototypes go through a production process, going from a working design to iterating on engineering design (electrical tracing etc.). The candidate design must be verified in anticipation of expected component tolerances and varying operating conditions.

A design for manufacturing, assembly & test (DFM/A/T) process is carried out to ensure viable production. Alongside DFM a manufacturing test jig must be designed and built in conjunction with DFM process for functional testing of every board. Each manufacturer builds the test jig in line with the electrical characteristics of the board, typically a bed of nails is created to test the electrical continuity of a pcb board, this ensures all netlist traces on a pcb that need to be connected are connected (see fig 2). During this process all boards need to be flashed with the products firmware intended for retail use, the common standard for flashing MCU's is JTAG (join test action group) protocol, JTAG is a 4 wire serial protocol, with a similar brother standard called SWD serial wire debug, these 2 protocols are commonly exposed on many pcb board as a way of flashing and debugging the boards. Mainly they consist of a clock and a data signal line for communication. The flashing process can be lengthy can requires manual flashing by placing a board on top of a programming fixture.

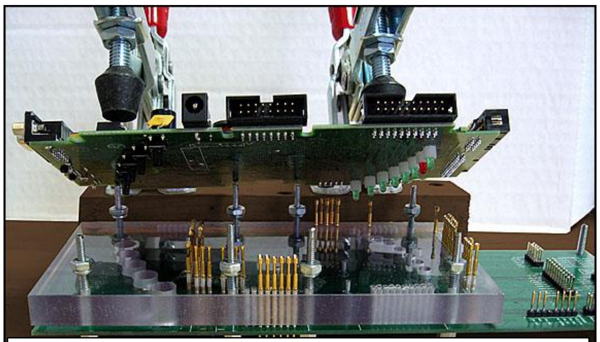


Figure 2- Bed of nails test fixture

At the end of step 3 a comprehensive product specification must be drawn up including documented quality and test measurements must be produced. Including component availability, obsolescent risks and supply chain fragility must be accessed. At step 4 a design validation test is carried out, test procedures are finalized and deployed by manufacturing partners. Production is initially slower at first to analyse yield and quality. Tests depend on nature of product, but all functional and aesthetic requirements of the product must be verified. This may then lead to future iteration in product or manufacturing design, for example an iteration in manufacturing design might be the size of the board, type of components used etc. to reduce cost.

At step 5 after DVT the product is ready for launch & a product validation test (pvt) which replicates reqs of DVT but during full speed production, first yield optimization is carried out which adapts production to maximise output quality and throughput. After 1000Pvt units are successfully manufactured the design and validation phases are complete and production moves to step 6 (regular production).

Software tooling has been built to facilitate quick iteration and testing of interactive devices to improve phase 2 such as calder toolkit & D.tools, interactive products can also be designed and prototyped AR/VR, but it is no where as developed and new user friendly compared to phase 1 prototyping a new device. [6] Emphasising that phase 2 activities of hardware production are limiting factor in adoption of new ideas. Scalability is key difference between hardware and software production, including difference in able to being able to produce replicability in devices, taking into account component tolerances in manufacturing, variability in manufacturing, sourcing consistent materials and other factors cause so many differences in copies of devices, emphasising issue of 'replication problem' [2] which is the issue we are trying to address which requires designing, constructing and using custom tooling.

Another issue in hardware development is iterative development, it's hard to get user feedback on early prototypes when they're still low on features, use feedback is only possible when prototype is nearly feature complete, at this stage it can be very costly to change the design due to potential changes in manufacturing tooling incurring more money and time lost in NRE (non-recurring engineering) phase of development. Updatability is another key issue incurring high costs requiring physical handling of every affected unit, shipping, re-work and testing of the unit. Therefore, all these extra costs make enabling hardware devices at a lower production volume (also known as long tail devices in hardware market, with short tail being small amount of mainstream popular devices and long tail being niche lower volume product), extremely challenging.

A way of enabling hardware devices as lower volume production includes making use of economies of scope [4], where efficiencies are gained from commonality across multiple products. This is an application of our project validate, where Jacdac is the commonality between products, all make use of similar components, tooling requirements reducing production cost with devices implementing Jacdac. However we still have the issue with engineering costs and tooling such as bed of nails test jig tooling which is made uniquely by each manufacturer. With the commonality of Jacdac, our solution validate empowers the commonality by making a standardised test jig and solution for any product that implements the Jacdac protocol. Creating a test harness that knows how to communicate with Jacdac devices, measure electrical characteristics to determine component validity and functionality. Currently there has been no previous research or standardised way of designing electrical device tests in parallel with functional testing procedure. An example has been made for wood working called JigFab [7], making a test jig out of standardised components.

Devices usually require and enclosure as well, inspection molding is most popular process for high quality enclosures (as laser cutting/3d printing has much poorer build quality) but adds high cost to NRE sometimes up to $10k even for small electronics, due to specialised tooling and material costs. Therefore supporting keeping NRE cost as low as possible is our goal.

Creators usually have a hard time being able to negotiate with manufacturing partners as well, some larger factories require <= 5000 order units to consider manufacturing your product, which is a high up front investment cost for start ups. Ideally we would like manufacturers to match operating scale of the creator but limited due to upfront costs and costs of investing in specialised tooling for their products. Therefore supporting commonality and Jacdac powered devices helps with making manufacturing more accessible for low volume products. Usual process for moving from protype to manufacturing is uploading design files (gerbers- pcb layout files) and BOM (bill of materials- component source and cost) to a manufacturer, but in addition you do need per unity functional testing with test jig and associated firmware adding cost. If you have not rigorously design for these steps errors can be missed.

Therefore building relationships with manufacturing partners early on is useful to optimize production workflow & cost ex partners can provide feedback on pcb board design, ex cost can be much cheaper based on just dimensions of the board! Therefore the goal with validate is to encourage creators to design for manufacturing during the prototyping phase, creating a test jig for your devices that any manufacturer and independently use and creating an efficient production and testing workflow. We create a pcb panel which lays our multiple devices on one large panel for cheaper production, less manufacturing waste and cheaper testing as we can design for automated testing as devices are interconnected on 1 pcb board.

The challenges in hardware manufacturing moving from prototype to product and how to solve them can be summed up as follows 1. Gaps in technical and non-technical knowledge can be solves with better documentation on the manufacturing process (guides, books and tutorials). Minimum viable rigor of a product and be solves with better version control & project management during product development (ex. Github for hardware designs- see changes in pcb among versions). Building relationships and professional networks can be solved with online communities and communications tools/hubs to connect with experienced creators and developing low volume production processes can include making standards for enclosure manufacturing and design patterns for manufacturing tests. Our goal with validate is to address the viable rigor of low volume products through creating a standard test jig reference for devices implementing Jacdac, ensuring rigorous tests can be developed during production as user has more control and making manufacturing much cheaper!

IV. Current methods of testing

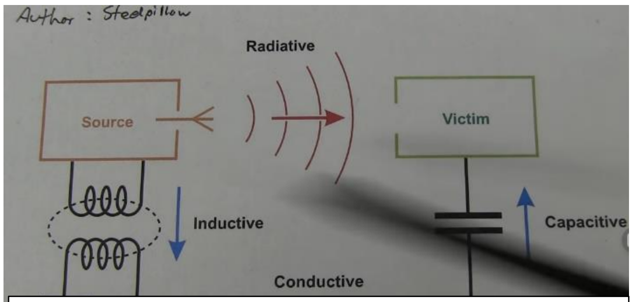


Figure 3 types of EMBC compliance tests

The current methods of automated testing include device under test (DUT) being connected to automated test equipment [8]. Typically equipment that injects signals/measures signals from DUT communicating with oscilloscopes, multimeters, signal generators, GPIO devices etc. JTAG is usually used to load firmware on DUT.

Test fixtures (jig) tests fabrication process worked and checks connected nets from components are connected and there are nets that aren't unintentionally connected, these are created from gerber files and netlists passed to the manufacturer, Automated optical inspection is also carried out on every PCB to ensure it's orientation is correct, solder quality is good, part number correct, to ensure correct assembly and mechanical integrity.

Electrical integrity testing involves testing impedance done with another bed of nails, ex. Was a correct value of resistor used in particular part of circuitry etc. Programming fiixtures are then used to program boards using a 'bed of nails' like jig to attach to jtag/swd headers using production programmers. Test routines are then carries out, example in firmware may require a special testing mode to support this and disabled before leaving factory. Example tests include flashing leds, sound buzzers etc. to ensure functional integrity of components, ex. You could use a colour sensing IC to ensure LEDs are shining correct colour.

Assemblers usually are responsible for doing components testing and packaging of each device for the creator, they usually charge per hour [8] for how long it takes to test and package a product, so testing efficiency and speed is very important for keeping price down. A way to address this covered by validate is the idea of testing panels (also known as embedded board arrays), where you can route and connect multiple embedded devices on one larger pcb, connect them to a testing connector (such as PCI-E or microbit style header or regular pin header) to test all the modules in parallel instead of one at a time for concurrent faster testing.

Pre-compliance testing also needs to be carried out for CE/FCC compliance certifications; this is to make sure each device isn't emitting dangerous electro magnetic radiation. An example is conductive emissions testing using a lisns device, other tests include capacitive and inductive but these are near field affects and only needed for specific products (see fig 3)**.** Conductive testing using a LISNs (line impedence stablisation network) device inserts in series with power cable going in, it couples off into a specturum analyser, the conductive filter provides a fixed impedence and source impedence from power is unknown.. This process requires a low impedence line so needs shorts grounding point. The spectrum analyser then measures radiation on the line, if the radiation is too high you can use components such as ferrite cores to limit radiation, but it's a good indicator early in design process whether your product needs to be redesigned before it becomes more costly post production once the product goes into compliance testing. Therefore compliance testing early in design is good for cost saving.

When it comes to production, the creator will create a BOM (bill of materials) containing all components/part numbers on their board, it's important they check if it's easily manufacturable by the supplier, you may need to buy more components than needed as components are shipped on reels for pick and place machines etc. therefore here is where economies of scope comes into play [2] it may be cheaper for ex. To place 4 2k resistors in series rather than investing in a 10k resistor if your board mainly comprises of 2k resistors etc. It's also important to supply aternative suppliers/qty per reel if particular ones go out of stock. Creators should also design a panel for their designed board for manufacturers to create, as it allows for a more efficient manufacturing process. A panel requiers extra tooling strips in your array of boards so it can support going into reels into manufacturing machinery. Panels also allow you to use active circuitry on the sacrificial part of them (the part you through away when you pop out each individual board), when you cutout boards you can do it 2 ways, eiher by V-grooving the pcb fibro glass which things the board around the edges of the device so it can easily be snapped out, or you can usue breakout tabs, which need wire clippers to break off, v grooving doesn't support tracing from the sacrificial part of the board to each individual module as the v groove will cut any active traces, but breakout tabs will support tracing from each module to sacrifical part to connect multiple modules together. Breakout tabs need to maximum 2mm wide. Panels also need a fidicucial mark for reference point of machinery to place the components on a larger panel. Also the tooling holes can be used to mounts for test fixtures to test each module in parallel.

Designing a testing routing for a panel scales with the product complexity, when designing a test jig you must consider the coverage, how to test every single feature of a device, UX, how to visualize/interpret test data, automation, fastest way to carry out tests & avoid human input. Audit and traceability, how to enforce testing standards, ie. Keep state of tests as they're running, how to not let devices slip through unproperly tested etc. Updates, how to maintain testing firmware up to date, responsibility who's responsible for product quality and how to encourage design for testing and code structure how to make testing process as efficient as possible.

Deriving tests is hard but can be split into 2 sections using inside/outside methodologies. First look inside and pick key features from the device, such as by looking at the schematic to test specific characteristics like solder faults on components/pins etc. Then you need to look outside from a perspective of a user usage, example testing function characterisitcs of a button, indicator led, screen etc. When updating a jig to support new tests it's important to test new firmware first on jig before pushing to all production jigs as it may break them, therefore it emphasises having version control in firmware versions for the test jig as well as the component firmware.

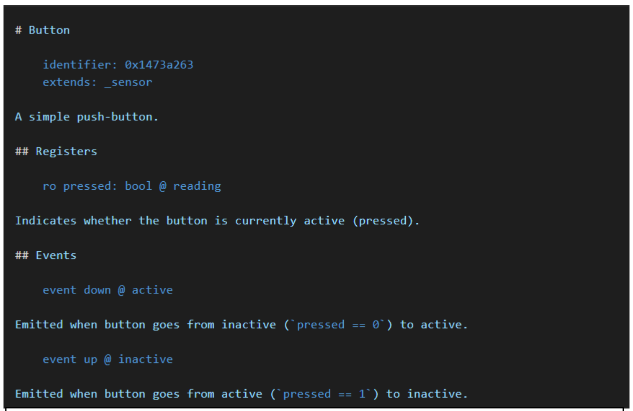
V. Validate design

Validate is a hardware verification DM tool for Jacdac components, checking whether hardware is meeting it's operating specifications, implementing JD service protocols correctly at 'physical layer', including time sensitive measurements on packet delivery, protocol implementation etc. Checking whether each module is operating within electrical spec, functioning within normal behaviour tolerances, and supporting the EVT, DVT & PVT processes in production to generate your own test requirements based on your module, but implementing Jacdac as a common foundation.

Validate addresses the sub-section of the replication challenge, encouraging standardised reusable test tooling for cheaper, more agile production, providing a new solution to small volume manufacturing processes, the test jig having in build interoperability. Making use of the idea of designing a re-usable test fixture (jig) for common hardware components (our focus being Jacdac), enabling cheaper tooling costs for manufacturers due to reusability between products sharing JD hardware components, making use of economies of scope opportunities at manufacturing hubs such as Shenzhen [2].

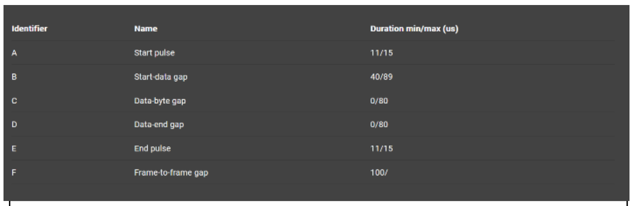
The types of testing include Jacdac service implementation testing, packet/UART serial timings, functional testing where device operates according to operating specification requirements- does LED light up, expected motor RPM of component etc. and it supports non-functional electrical diagnostic testing, measuring current draw during functional operating usage, voltage tolerances, testing voltages the device can operate at if you limit the voltage supplied to it, does it meet that listed in the spec? and netlist continuity testing, by checking solder paste by being able to check each connected pin through functionality testing. An example would be to test LEDs on an array of embedded devices on a panel by designing a test fixture to mount onto our pcb panel, the test fixture will have ac/decouple photodetectors to detect whether LED is shining on each individual board.

First let's discuss Jacdac, Jacdac is a newly developed bus-based plug and play hardware/software stack for microcontrollers and their peripherals (sensors/actuators) that supports rapid prototyping, making and physical computing devices [9]. Packets



```
# Button

    identifier: 0x1473a263
    extends: _sensor

A simple push-button.

## Registers

    ro pressed: bool @ reading

Indicates whether the button is currently active (pressed).

## Events

    event down @ active

Emitted when button goes from inactive (`pressed == 0`) to active.

    event up @ inactive

Emitted when button goes from active (`pressed == 1`) to inactive.
```

Figure 4 Jacdac Button service



Identifier	Name	Duration min/max (us)
A	Start pulse	11/15
B	Start-data gap	40/89
C	Data-byte gap	0/80
D	Data-end gap	0/80
E	End pulse	11/15
F	Frame-to-frame gap	100/

Figure 5 Jacdac service timings

are sent serially among physical devices on a jacdac bus and sent over UART TX (half-duplex), so you can only send and receive data at separate times as only 1 device can have control of the Jacdac bus to send data at any given time. The protocol sends data packets by transmitting 10 bits at a time, 1 start bit, 8 data bits and 1 stop bit. Transmitting at 1Mbaud bits per second. Jacdac packets consist of 3 types including registers, commands and events which together creates a service, are all defined per device and written in plain text mark down to define a device service. (see figure 4 of a button service as an example) The button service sepecifies a service identifier which is how it's uniquely identified on the jacdac bus, registers for advertising which data can be read or written to the service and events for actions happening on the service, ex if a button is pressed or not.

For a jacdac device to take control of a bus it must implement the follow timings to take control, send data and to release the bus (see figure 5/6). The start pulse consists of pulling the jacdac bus low for 11-15ms, recognised as a break condition in most uart hardware, when the start pulse is over devices have 50ms to configure IO registeres to receive UART data, if n o byte is received within 200ms an error condition is raised and devices have to wait for bus to return to idle state. Meaning first data byte must be sent at 189ms after completion of start pulse (ideally after

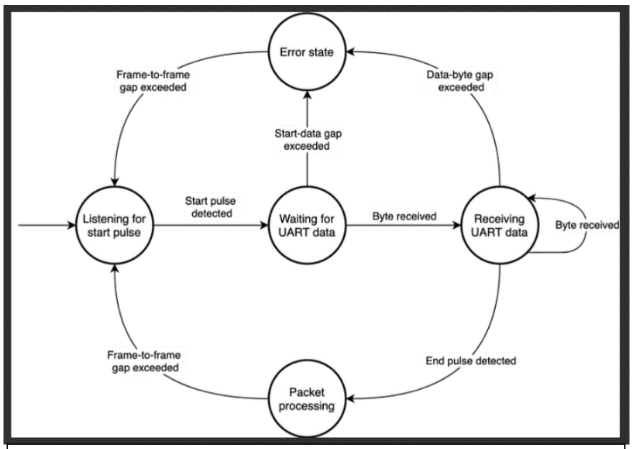


Figure 6 Jacdac service routine

50ms). Once a device has sent it's data it must signal an end pulse which is a low line for another 11-15ms to trigger another uart break condition, and must occur within 80ms of the last data byte otherwise an error condition is raised on the bus. Devices then wait 100ms + rnd time value per device before sending again (this is so all devices don't send and try to take control of the bus at the same time).

VI. Validate Architecture

The validate testing architecture comprises of a front end website acting as a test runner running typescript and react which communicates with an intermediary Jacdac bus the jacdapter which connects to a panel of Jacdac modules to the Jacdac test jig. The Jacdac test jig is a Jacdac device I prototypes and developed which is responsible for carrying out testing services on connected Jacdac modules on a panel, it has exposed gpios for external testing components to run on connected modules, this adds interoperability as you can switch out testing components on the test hat as you like depending on the Jacdac device you are testing, the envisioned idea is that a test fixture will connect to the Jacdac test hat and will be controlled by it to run the tests. The Jacdac test hat also is responsible for measuring the current draw by each individual module, each test the testing hat carries out is defined and specified in the Jacdac spectool and written in plain text called JDOM (similar to how a service is defined). For example figure 7 shows the temperature module tests the test hat runs in JDOM, which is interpreted and displayed on the front end website. JDOM is acting as an abstract testing specification language to write tests as a novice user to abstract over fast technical details.

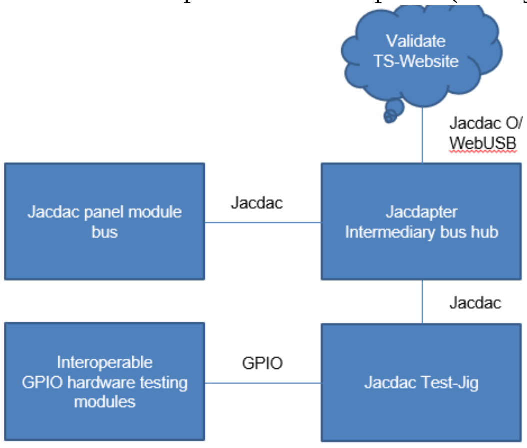
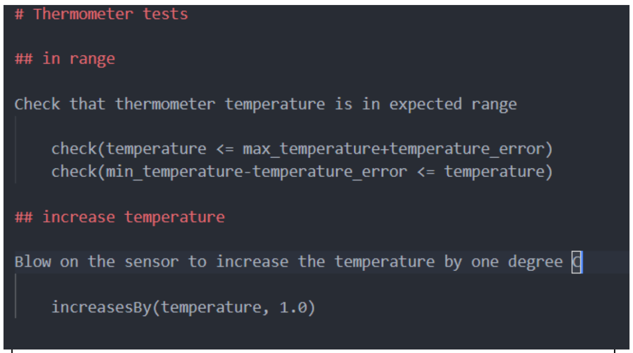
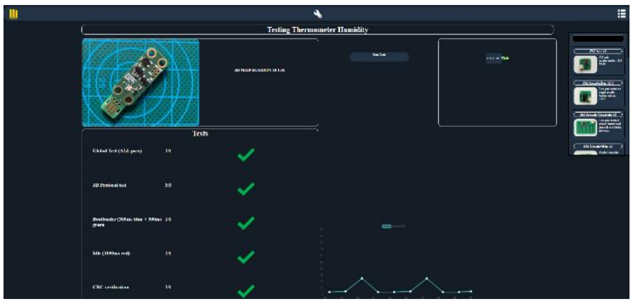


Figure 7 Jacdac Validate testing architecture



```
# Thermometer tests

## in range

Check that thermometer temperature is in expected range

    check(temperature <= max_temperature+temperature_error)
    check(min_temperature-temperature_error <= temperature)

## increase temperature

Blow on the sensor to increase the temperature by one degree C

    increasesBy(temperature, 1.0)
```

Figure 8 JDOM specification of tests run on temperature module



Testing Thermometer Humidity
Tests
Global Test (ALL pass) 1/1
JD Protocol test 3/3
Bootloader (500ms blue + 500ms green) 1/1
Idle (1000ms red) 1/1
CRC verification 1/1

Figure 9 Non-Functional testing of Temperature module on dashboard, testing firmware 0.14

The front end testing dashboard displays connected Jacdac modules visible over the Jacdac bus on the panel. Shows the status of the Jacdac test-jig, and shows selectable tests to run under each module.

The testrunner takes defined service tests from JDOM markdown/json specifications and implements an execution test suite for this. The website makes use of Jacdac-TS implementing JDOM object model library for communciatrion JACDAC to each devices from the website to the modules over UART from a connected microbit (jacdapter) to your pc to the panel of Jacdac devices.

The test hat (jig) itself is commanded by the website to interact with the jd devices on the connect panel on the jacdapter to execute tests. It has in built interoperability controlling GPIO testing components for specific test cases, etx. Rotation sensor, photosensor, LDR etc. the test hat acts as a baseline test jig with interoperability built in, tests defined are user specific in markdown. The baseline Jig functionality is responsible for testing current and voltage measurements along the JD bus for each panel device. The website can also flash specific module firmware to each Jacdac module and ensure correct JD protocol implementation. Every time you release a new Jacdac firmware for your module you can use

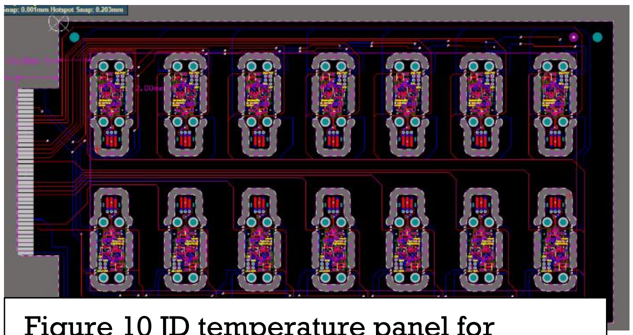


Figure 10 JD temperature panel for simultaneous testing

validate to test if the firmware conforms to normal operating standards, including physical Jacdac standards, and operating standards of the device to make sure there is no unknown hardware issues with designed firmware that cannot be shown by a regular compiler.

Figure 9 shows the website in action showing the testing of the temperature module testing the JD service protocol implementation of the module along with whether it's operating within it's expected electrical characteristics, you can also choose to test multiple versions of the devices firmware on the website, therefore adding the layer of CI/CD to hardware development, as whenever you release a new firmware version for a module you can check whether it affects the non-functional and functional characteristics of the devices hardware as well. Thereby adding an extra layer of software CI/CD by integrating hardware testing for the software as well. This will really help developers as you can test your hardware as you prototype instead of finding hardware issues at a later stage of development during manufacturing, reducing NRE cost and making device production more cost effective.

The following also demonstrates my designed panel for parallel module testing, using the dashboard, it will automatically detect all the temperature modules connected on the bus and run the tests simultaneously on each individual module as the Jacdac bus is routed to each temperature module and is connected. It also shows how we're 'bit banging' swd to program all modules from the validate website as well, it makes use of the standard microbit male header, exposes the swd pins and we can control each pin to program each individual module.

VII. Conclusion

Therefore with my validate solution we can easily control each module from our testing website, being able to individual program each module, adapt to testing different kinds of modules requiring different testing hardware due to interoperability of the test hat, you can connect any testing components you like. You can test new firmware versions and flash it using validate to ensure software validity as well as hardware validity to new firmware, adding a layer of CI/CD that will save cost and development time in early prototyping as well as future production and hopefully encourage the long tail.

Bibliography

[1] Microsoft, "Democratizing Hardware Initiative," 1 July 2019. [Online]. Available: https://www.microsoft.com/en-us/research/project/long-tail-hardware/.

[2] S. Hodges, "Democratizing the Production of Interactive Hardware," in *UIST*, 2020.

[3] Microsoft, "JACDAC," Microsoft, 2021. [Online]. Available: https://microsoft.github.io/jacdac-docs/. [Accessed 17 March 2021].

[4] N. C. Steve Hodges, "Long Tail Hardware: Turning Device Concepts Into Viable Low Volume Products," in *IEEE Pervasive Computing*, 2019.

[5] S. H. Rushil Khurana, "Beyond the Prototype: Understanding the Challenge of Scaling Hardware Device Production," in *CHI*, Honolulu, 2020.

[6] "D.Tools," [Online]. Available: https://www.d-tools.com/. [Accessed 04 06 2021].

[7] JigFab, [Online]. Available: https://hackaday.com/2019/04/22/jigfab-makes-woodworking-easier/. [Accessed 04 06 2021].

[8] D. Cad, eevblog, [Online]. Available: https://www.eevblog.com/. [Accessed 04 06 2021].

[9] "Jacdac," Microsoft, [Online]. Available: https://microsoft.github.io/jacdac-docs/. [Accessed 04 06 2021].

[10] Microsoft, "JACDAC Dashboard," January 2021. [Online]. Available: https://microsoft.github.io/jacdac-docs/dashboard/. [Accessed 18 March 2021].