CPS E20 Jim Mussared

Sun, Oct 02, 2022 6:57PM 22:27

SUMMARY KEYWORDS

micro python, python, project, damien, pcb, micro, electronics, async io, run, bytecode, features, circuit, board, cases, bytes, async, damian, microcontroller, rework, add

SPEAKERS

Paul Cutler, Jim Mussared

P **Paul Cutler** 00:02

Welcome to the CircuitPython Show. I'm your host Paul Cutler. This episode I'm joined by Jim Mussared. Jim is a maintainer on the Micro Python project and works for George Robotics. He's a father of two part time students and lives in Sydney, Australia. This episode is brought to you by PCB way. With over a decade of experience. PCBWay is one of the most experienced manufacturers in PCB prototyping and design. Whether you're an engineer students or hobbyist PCB way offers a simple and fast prototyping service and it's cost effective at only $5 for 10. PCBs in check out a PCB way.com/project Where PCB way helps makers and hobbyists collaborate on their designs and projects. Make your design a reality and check out PCB way.com For all your PCB needs. And they also now offer CNC machining and 3d printing services. Visit pcbway.com for more information, thanks to PCBWay for their sponsorship. Jim, welcome to the show.

J **Jim Mussared** 00:59

Thanks, Paul. It's really exciting to be here.

P **Paul Cutler** 01:01

I'm glad you're here. How did you first get started with computers and electronics.

J **Jim Mussared** 01:05

So I was always interested in that side of things. One of the one of the first memories I have of being really excited about electronics is as a member of a country that also says, Python, not Python, like Nicolas Talavera, in the earlier episode, I was given access to a BBC Micro as a as a very small child. And there was an amazing system that you can interact with Lego motors and sensors through. And it's pretty much I was very grateful for one of my teachers who took a special interest in that and gave me a lot of time to play with that. And so one thing led to

another I studied electrical engineering at university, but went off into software did process control, and like software operations. And then much later, through a couple of different avenues, I was reunited my with electronics. One in particular was I got quite into sailing, like like racing. And if you want to race it a, you know, moderately high level, you either need to be an excellent sailor, which I am not, or you need to have some sort of unique skill to add to a boat. And it turns out, there's an awful lot of electronics in highperformance sailing, we set sail by numbers is kind of the joke, and being able to quickly repair and configure and get the most out of electronics. It's quite, quite useful. And through that I got really into repair and reverse engineering and future rabbit hole in YouTube of the amazing resources that are available to really get excited about electronics on YouTube. And, you know, you talk to your friends who, you know, what did you do last night, you know, watch this on TV or whatever it's, Oh, watch the three hour episode of some Scottish guy named Ian repairing an eight digit multimeter it was amazing. And, and then through that I've always been interested in education as well. And I got an incredible opportunity to teach at a summer school in Australia for high school kids for several years, and we taught electronics and circuit design and and through that I got involved in a company that specifically taught Python to to school kids. And with that got very involved in the BBC Micro bit. And through that I met Damien at Pycon PyCon. Au, which is a wonderful spin off of the PyCon conference in Australia.

P Paul Cutler 03:25

And Damien George is the founder of micro Python, how did he first start that project?

03:31

So in 2013, there was a Kickstarter for the original Pi board. And it was a combination of this electronics component and and the software component to run on it. My understanding I wasn't involved at all and I was didn't participate in the original Kickstarter. But my understanding is that this kind of started as almost like a dare sort of a bit you can't make Python run on an STM 32 or four. And now that I've gotten to know Damien really well, this title, this story totally fits. And the Kickstarter was was actually quite successful, successful and they followed it up with another one micro Python to the ESP 32 C six. And I always joke to Damian every time we have a crazy idea for a hardware project. Oh, we should do another one another Kickstarter is net. Never again. To was enough. Yeah, absolutely.

P Paul Cutler 04:18

So it's been almost 10 years since micro Python and the first Kickstarter, and it's been about five years and circuit Python actually forked micro Python. What are some of the differences between the projects?

04:30

So much of this happened before I was involved? really actively in the project I was I was actually off on the side doing the the Grok Learning micro bit. So I think it's really interesting to consider the idea of Python on microcontrollers as this incredible, I would say configuration

space of different options, options for how you'd want to proceed with a project and what you want available to you and how to do it. And I think it's really actually wonderful that we have two very different solutions that cover much widest or arrangement of configurations and interests and, and in particular that the two projects share the core. I'm really, really excited that in the last a year or so, so for Python have done a huge amount of work to get the core in sync effectively. And micro Python is able to specialize on things that make sense to micro Python and circuit Python is able to specialize on on use cases, it makes sense to separate Python. And there are lots of things that are in circuit Python that I look at and say, oh, yeah, that, that totally makes sense. We're not gonna do it that way. Because we have to support this other use case that's not compatible. Interrupts is probably a good example there. But and so Python, as others have mentioned before, very much built a workflow around copying files to the device through the through mass storage. We work really closely with with Adafruit. I talked to Scott pretty often and more recently with Jeff and Dan as well. We're hugely grateful to Adafruit for sponsoring the micro Python project, and doing a lot of behind the scenes stuff that that just really helped the brand and the project in general. Lady Ada and Phil have been tremendously supportive.

P Paul Cutler 06:16

What are some of the areas that you've had the opportunity

06:22

it's very difficult to design a system that works one way for one person and one way for another person. And circuitpython provides a whole different range of use cases that would that we would never think of, and we don't see. And so we we get quite a lot of really useful feedback from from Zephyr Bethan team that influence how we design features, we often talk to them before starting a piece of work and, and we get bug bug reports and, and pull requests straight that we try and move straight away up into upstream. We had a great example recently where Jeff had an idea for reducing the firmware size and then we took that into into micro Python is really, really useful.

P Paul Cutler 07:08

What do you see as some of the strengths of micro Python specifically?

07:13

I think one of the strengths of micro Python is that it is much more building blocks and configurable to a particular use case. And this comes up a lot more in our professional uses a micro Python where we're doing very custom implementations of the core VM for a particular customer or a very specific board configurations. A lot of these customers really want to use very low level features of peripherals and things like that. And so we're, we need to expose much lower level details in the API's and they need interrupts and low latency. And

P Paul Cutler 07:54

along with Damien, you're one of the maintainers of micro Python, what's it like to be a maintainer, especially if such a large project

08:01

so I was quite new to open source when I when I got involved in the micro Python project as it as somebody is used a lot of a lot of open source software, but but actually contributing and working on open source was new. It's a tough job. You're pulled in hundreds of different directions from everybody, everybody's little use case to and it's very hard to figure out what's important and what's worth prioritizing. And because ideally, you'd like to do everything, it's and especially someone like me, I'm I'm easily distracted by Well, yeah, we totally should fix that right now. And maybe that's not the highest priority. The other thing that's hard is that we have so much stake in our heads as maintainers in terms of where we would like to go, but it's not really quite fully crystallized and, and coming up with a good way to get the most out of the community to have them contribute in the most efficient way possible. Realize it requires quite a lot of time. Documenting and explaining and and that can be really hard when you haven't quite figured out the direction you want to go in anyway. But on the other hand, it it is a really exciting job seeing all of these people come together from a wider range of different use cases, we've got people running micro Python, on Windows in scientific systems to people using really really tiny micros to people using the Unix port on quite big systems and micro Python in space and micro Python and toys and it it's it's an exciting thing to see all this come together and in a system that works and and overall maintains really high test coverage and code quality and, and and efficiency and code size. It's it's really fun to get has completely transformed the way we do software development. And if I could go back 10 years and only highlight a couple of things that are different in the industry 10 years from then that would definitely be one of them. We also, as a maintainer of the open source project, we have some challenges involved in how we also mix that time with the professional work. So Damian into lesser extent, me to do consulting work with micro Python for professional clients. And so finding a good balance between the features that they want and the features of the open source project one, and ideally funneling as much of that back into the open source project as well.

P Paul Cutler 10:23

For you, and Damien, how do you find time to balance both your professional work and the open source needs of the community?

10:32

It really helps. In terms of Damien, I communicating that we are good friends, we have kids about the same age. Unfortunately, Damian lives in Melbourne, I live in Sydney. But we find it very easy to talk and plan and have fun in that process of running the project. And I'm a part time student, I have much less time to put into this than Damien does. And Damien works really, really, really hard. And I think that's the answer to your question is Damien just works really, really hard. Often, often, the best results come when we can align the professional work with the open source work, and what the ideal there is where we have our professional

customers actually contributing pull requests and raising issues and basically contributing to the project as if they were open sources. And we, we do have that quite often. And it's fantastic. Oh, that

P Paul Cutler 11:27

is great to see. You mentioned some of the interesting use cases for micro Python, what are some of the favorite ones that you've come across in your years of working with micro Python.

11:36

So at the opening, I'd said that I got excited about electronics through the BBC, micro and Lego. And every day, it's it's just amazes me that Lego now use micro Python in their Mindstorms products. And the fact that the successor to the BBC Micro, the micro bit, also runs micro Python. And that makes me very happy. There's a whole range of different things, we do some really cool work in Australia for a company that does medical devices that use micro Python. They use micro Python as their primary platform for development of, of these embedded devices. And it's amazing working with them, because they really push what we can do with with micro Python. There's a great talk, you can look up from Damian from PyCon au from a few years ago, about his work with the European Space Agency for space qualifying micro them. So the mission hasn't flown yet, but that work is still relevant and exciting. I've seen micro Python used in bench equipment. And there's a recent example I saw just chatting to him the other day, and Robin, who makes the pixel pump as a Crowd Supply project. There's a company in Germany that have used micro Python for car counting machines on the autobahn. I was at at at university that day, and somebody showed me their calculator and said, Oh, my calculator runs Python. And I'm like, for real and it ran micro Python as it turned out. And so we even get surprised by some of the places that run micro Python. Yeah, it's it's, it's, we have this reputation that Python is slow, or Python is bloated or any of these things. But the reality is that in a lot of these projects, what matters most is developer productivity. And also being able to build products that are easily testable, and adaptive to different board and hardware configurations. And right now, being able to take out because you can't buy STM, 30 turns or whatever, being able to replace the entire microcontroller. And within a day's effort, keep running your existing firmware. And so yeah, there's there's there's amazing opportunities for micro Python, I think in this in in, in this space.

P Paul Cutler 14:02

Micro Python 1.19 was just released this past June and saw significant performance increases with the way mpy files were reworked. When you're dealing with micro Python and microcontrollers, performance is always at a premium. Can you tell me about some of that work that you did?

14:20

It it seems really strange to me coming from a previous job where I literally worked in exabytes. And and you know, we would joke, you know, what's that number you just said, zero petabytes, and and to come down to this world where bytes matter in you know, individual individual

and and to come down to this world where bytes matter in you know, individual individual bytes. We go to great lengths and commit extreme crimes against the C compiler to save 10 Bytes here and 20 Bytes there. But it adds up. And something like I just looked this up before the call was F strings the entire feature is 500 bytes. And so if we can rearrange them structure or figured out a way to rework a few functions and change the inlining. We can offset that. And our goal is to keep the base core of micro Python almost constant and continually add new features. It's extremely difficult because different architectures and different configurations will, maybe the same optimization makes one smaller than the other bigger. But it's actually really fun. Some people like to make chips in a bottle, and we like making our code as small as possible, ideally, without making it also really ugly and difficult to work on. Kind of related, sometimes increasing performance will also decrease code size, but often, they go against each other. So we're always sitting there with the performance tests running, I have an automated thing that runs on every pull requests that tells us the performance destructive across a bunch of different boards. And when we can find the changes that do both, it's fantastic. I think it was actually not 119. But 118, we put in a feature that actually isn't quite interesting history that something that micro Python implemented originally, which is to cache in the bytecode. When you look up something, you know, map, which happens all the time, every time you access an attribute on an object, and location in the bytecode, where that attribute was found. So that the next time it's faster. And then Python itself added that in C Python, and unfortunately for us, we never got much value out of this feature, because most of the time our bytecode is in ROM and so we can't modify the bytecode. So we found a way to implement that casing in a kind of a lookaside buffer, and realize the same benefit in micro Python. And that was really exciting, this 20% performance boost on some boards and even higher on others. You mentioned the NPY rework. One of the things here is we're constantly looking for ways to move. data out of RAM and into ROM chips typically have a lot more RAM than they have RAM. And I'm using Rami a bit confusing, I mean flash. And what this will let us do and I hope circuit Python and maybe we'll see some benefit from this, too is have a way to dynamically update code on the board to Python code on the board. But not have to have that code run from RAM have it run from flash instead freeing up significant amounts of RAM for the for the programs to run. And, you know, we might only be talking small, small amounts of RAM in an absolute sense. But if you've got a program that's constantly hitting around limit, adding an extra kilobyte, makes all the difference. So it's a pretty exciting feature. And hopefully in the next couple of releases, we'll start to see that all come together.

P Paul Cutler 17:50

So where else do you see micro Python going next?

17:55

We have a lot to learn from circuit Python, I think. And that's one of the things that I really want to focus on for micro Python community documentation. making it really easy to get started with micro Python. One of the things that tie back to what I said earlier is that people come to us and they want to figure out how to configure micro Python in in crazy ways that that are really unique to their situation. And I want to make that really easy.

18:25

There are a never ending series of Python features that we need to consider implementing in

There are a never ending series of Python features that we need to consider implementing in the core. And also a never ending series of chips and architectures that we need to add support for. Another area I'm particularly excited about is async IO. So for the last few releases, we've been adding more and more support for for async. IO. And having now written a few pieces of microwave and software that that use async IO it is it is transformative. Originally circuit Python didn't have interrupts because they are hard to use. And micro Python has interrupts and they are hard to use and lead to lots of problems. And I feel like async IO gives you that balance of being able to write asynchronous code that is still maintainable and understandable and efficient. So we would like to see a scenario pushed further and further into the core. I want to be able to await on a pin changing state or a weight on a DMA transfer. And, and use that everywhere. One of my favorite examples of this is using async IO with Bluetooth, which was extremely difficult to do when you have a complicated protocol that involves callbacks backwards and forwards and, and now it's all just straight line linear await sequences.

P Paul Cutler 19:44

There's definitely a lot of excitement around async IO I think in both communities.

19:50

It's difficult that there's a lot of different ways to do the idea that is async Oh, that's true and, and so forth. I think pragmatically we need to just focus on choosing one. And, and I use cases are simpler than what people are doing in C Python. And I'd like to just to see my I think I'd like to see, I would like to see async IO work really well and, and cover as many use cases as possible.

P Paul Cutler 20:18

Well, we're almost out of time. But before we go, I have one last question. I'd like to ask each guest. You're about to start a new project or build a prototype, which microcontroller do you reach for? So,

20:29

this I've enjoyed this question on on previous episodes, and I always laugh because in our work, we don't use the microcontroller we come in with somebody else has already chosen it. And the idea of choosing is a luxury look, as an employee of Georgia robotics, purveyors of fine micro Python products, the Pi board is my is my obvious answer. It is it is a wonderful board. I've used it in many of my own projects, I really liked the way it's designed and the way the pins are laid out. But of course, I have to say that I have a very fine, large. I have a huge fondness for the micro bit. And I've enjoyed already using the micro bit with my son, he got a remote control car for him recently, and the controller was too big to fit in his hand. And one of the amazing features of the microbead is this packet radio, which is that it's the Bluetooth, Fi but without bluetooth, Bluetooth Low Energy and limited on top of it. And you could just use it to send arbitrary messages with no stack and no knowing how to do anything. And we built a remote controller for his car that he could use the accelerometer to determine and steer. And I love

using the micro bit for that reason. But on a personal note, if I were to do another project, I would want to do something with an FPGA. And just because I always love chips that have crazy peripherals. And so I think my next project will have a nice 40 minute reading Mark.

P Paul Cutler 21:53

Nice. Great fig. Jim, thanks so much for being on the show.

21:57

Thank you for all it was heaps fun.

P Paul Cutler 22:01

To learn more about micro Python does it micro python.org consider supporting them financially via GitHub sponsors. Thank you to PCBWay for sponsoring this episode of the circuit Python show. Make your designs a reality with 10 PCB starting at only $5 Thank you for listening for show notes and transcripts visit circuit Python show.com. Until next episode, stay positive