cps-e22

Wed, Nov 09, 2022 9:14AM 20:38

SUMMARY KEYWORDS

gadget, python, circuit, lcd, adafruit, book, squared, build, battery, deep sleep mode, joey, arduino, pcb, feather, power, objects, drawing, started, prototyping, project

SPEAKERS

Paul Cutler, Joey Castillo

P **Paul Cutler** 00:02

Welcome to the CircuitPython Show. I'm your host Paul Cutler. This episode I'm joined by Joey Castillo. Joey is a hardware and software engineer based out of San Antonio, Texas. In 2019, Joey began branching out into hardware and Electrical Engineering. He created oddly specific objects, a division dedicated to designing, manufacturing and selling comprehensible open source gadgets for hobbyists and creative technologists. This episode is brought to you by PCBWay. 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 but only $5 for 10 PCBs and check out 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. Joey Welcome to the show.

J **Joey Castillo** 01:11

How are you? Good to be here.

P **Paul Cutler** 01:13

Tell me how you got started with computers and electronics.

J **Joey Castillo** 01:16

So I guess how'd you get started computers and electronics it starts way back. I mean, I was like into computers as a as a kid like, I don't know, like my my, my aunt gave me her compact portable from like the 1980s when I was in like grade school, and I started playing with that also tried soldering things together as a kid, but never really got very far. In more recent years,

I got into software engineering, doing kind of iOS apps, the App World app land kind of stuff. And that's from there, I started kind of hacking back again on electronics projects back in like 2019 or so, and kind of started to really make a go of it. And now I'm doing this office specific objects thing.

P Paul Cutler 01:53

So tell me a little bit about Oddly Specific Objects.

J Joey Castillo 01:57

I started in 2019, with the kind of the chaotic, exotic if you had announced that word, dream of building an open source ebook reader. And at the time, like I didn't know enough to know how to build that. So I started up with some simple projects. And my first project I called it the hiking log, it was a 3d printed log. And inside it was a feather and a GPS wig and a temperature sensor. And I could throw it in my backpack when I go on hikes. And it would log my location and temperature data when I go camping. I thought to myself, like it's a data logger that's shaped like a log that is an oddly specific object. And that was where the name kind of came from. Since then, I've started you know, building more of these kind of gadgets for sale. So I have the sensor watch, obviously, and then the LCD FeatherWing. Yeah, oddly specific object just feels like a you know, it's kind of like the box into which I put those kinds of projects more or less.

P Paul Cutler 02:47

I wanted to ask you about the LCD FeatherWing. You recently tweeted about implementing the ITC driver with circuitpython first before Arduino. Tell me more about the LCD FeatherWing. And what led you to that decision?

J Joey Castillo 02:58

See, so the LCD FeatherWing. It's a kind of a simple gadget. I mean, it's making it a little complex. But uh, essentially, it's an LC ra LCD glass, which requires Dr. Drew being driven by some kind of fancy analog signals. And chip that translates between the I squared C protocol that we all know and love. And these kinds of very analog kind of LCD driving signals. When you bring up a new thing. There's, there's all kinds of things that can go wrong. It's like, did I first of all, did I wire it correctly? Did I like mix up my TX and RX lines? You know, did I did everything powered properly, am I sending the right bytes over the bus to initialize it, and since you know, just put something on the screen. And when you're working on an Arduino world, you can call begin on the I squared C bus and you can send some bytes, but you're kind of just keeping your fingers crossed. Whereas with circuit Python, you know, if you forget to have a pull up on your I squared C line, you're going to get a message in the console telling you that you did that you made that mistake. So when it came time to write the I squared C Driver, there were a couple of factors that led me to go circuitpython First, first of all, it was just really easy to get started and make sure that I had it wired up correctly. I had one thing where I was like, Why isn't this working? Well, oh, it's there's no device at that address. Well, circuitpython told me there's no

advice in that address. And I went back and checked my datasheet and put the correct address in. But that's the kind of thing that could have had me flummoxed in Arduino land for a while until I like scan the whole bus. But seeing a really simple like error message telling me this is what's wrong, meant that I could get it right quickly. And I guess that's the second thing, fast iteration cycles if I was doing these character mappings where it's like, which segment maps to which position in a byte. So if I have something wrong there in Arduino sketch, I go in and I edit my bit masks, and then I compile and I upload. Whereas with circuitpython, if I change that bitmask and I just hit the save button, the code automatically restarts and I'm in business that really like super fast iteration cycle of working in certain Python really made an obvious first choice for developing the driver. Like I want to split second Python users, of course, but it's actually much easier for me as a A builder of gadgets to get started there. And then when it came time to do the Arduino driver, I more or less just started with my Python code copied into a C++ file and started just porting over line by line what I had done. But all of the hard part all the thinkI part was already done by passing Joey in Python. So that was kind of the the thought process behind building the circuit Python driver first, and then the Arduino driver.

P Paul Cutler 05:27

That's been a common theme on the show is that faster iteration cycle in that immediate feedback you get from error messages?

J Joey Castillo 05:34

Yeah, no, it's just really great. And like, again, like, it's the kind of thing that could have you flummoxed? Because you start out with this thing of you have, like all of these possibilities that it could the issue could be, and the faster you can get narrowed down to what it might be to what it is, the sooner you can address that and find the next inevitable issue. And you know, you knock over enough issues, and then you're done.

P Paul Cutler 05:53

You mentioned the open book earlier, which is an open source e reader using a Raspberry Pi Pico, how did you develop the user interface for an ebook reader?

J Joey Castillo 06:02

I mentioned in the beginning, I kind of started in like AP World and iOS app world. And so in the beginning, like at the beginning, I used UI Kit, which is this really cool kind of event driven system. And I still find it really intuitive. Like if you have like a window, okay, so you have a window, and then into that window, you add a table and in that table, you have a bunch of cells, right? If you wanted to lay something like that out, you wouldn't want to work in raw screencasts for that, right. So you have the window and you add the table as a sub view window, and you add the cells as a sub view of the table. And the label is a sub view of the cells. And everything kind of just lays out recursively. And it like just works. So when I wanted to do the open book, I really wanted to have a user interface framework like that, that would let me like kind of lay things out in an intuitive way. And this was another thing where I actually

got started with that insert iPhone, because, you know, eventually, when I wrote this in C or C++, most of my time was spent wrestling with pointers and smart pointers and weak pointers and shared pointers. Whereas in circuit Python, I can just like, define my objects and define the way I want my objects to relate to other objects. And in this case, I kind of piggybacked off of display IO, because my views in my user interface hierarchy are actually just groups with some extra like stuff sprinkled on. So once I started to do that, it was very easy to see like, Okay, now my layout is working recursively. Now I want to have a tap, and I want to see where that tap hit. So I can like, start with a screen coordinate, and recursively work my way backwards to figure out which object on the screen receive that tap. And then that kind of like bubbles up throughout the view hierarchy, to the point where like, okay, my cell can have a chance to respond to that tap, and then my table can have a chance to respond to that tap, and then window can have a chance to respond to that tap. That's a really kind of a complex system to kind of build up from scratch. But starting in Python world, it was, I mean, it was fun, it was actually really like quite easy to iterate quickly on that. Not only that, but once the framework was kind of built, I was able to easily like use some of the circuit Python libraries that existed to build cool stuff on top of it, whether it was you know, taking just sort of the like, you know, Adafruit has like a little less circuit Python characters, like the little resistor named Mo, and all that, you know, the characters, making a little table with those characters, and using a D pad to navigate between them, and tapping on one and popping up an alert. Like that was just a very simple way to like, demo something out, using just like the Adafruit display shapes library and the Adafruit image loader. But then I think it was Jeff Epler, did a mp3 player in circuit Python. And so I was like, Cool, I'll take the video player. And I'll port it over to circuit pi UI, which is the name I had for this UI framework. And in like an afternoon, I was able to put together a working mp3 player using the buttons and sliders and progress bars that I built in circuit pie UI. So it ends up just again, let's just to kind of a really fast iteration cycle, I was able to build cool stuff that I actually like playing with, but also it kind of exposed some of the like cracks or, you know, challenging areas, or like those rough edges that need to descend off of circuit pi UI. And that, like, ended up giving me like a very robust, kind of like, event driven, you know, recursively laid out kind of UI system. At one time, I actually did think I was going to build the entire open book, UI in circuit Python, for a couple of reasons that didn't like work out exactly as I hoped. But when it came time to build the whole thing, and C++, once again, just kind of like my taxi driver, I've got this really like, well thought out like framework that I've built in Python, and I've done it in this high level language that takes care of a lot of that, you know, stuff for me like, Oh, my views gone out of scope. It's garbage collected at some point and it goes away. Now it's up to me a little bit to manage that scope and to figure out whether it pointers weak or how, who has to hold on to what, but the hard part of like, how should it work? That's all something I got to think through in high level language like Python, which is yet huge. So

P Paul Cutler 09:56

what's next for the open book?

J Joey Castillo 09:57

So essentially, the open book is I've got This Raspberry Pi pico driven version of the open book, which is really finally, easy to hand assemble like this has been my main thing that the open book has always been the vision was a do it yourself ebook reader that someone can build from scratch and understand. And now I've got it down to a small number of parts all relatively large,

I'm going to bring 20 of the open book kits to super con in California to hopefully get people out there and have also published all of the design files. So folks can order the parts and build it themselves. If they want to get 20 of them and make it a project that they're hackerspace, they could do that. If they wanted to get 100 of them and sell the kits in a country where I can't sell things easily, like have that like steal this book. That's kind of phase one, I'm kind of calling this version the like developer preview edition, because it's kind of got chunky double A or triple A batteries sticking out the back. So it's not as spelt as a lithium polymer battery oriented solution might be. But that also means the park count is lower. And it's easier to deal with shipping and transportation and all that. So yeah, what's next for the open book is the kind of Developer Edition and then we'll see next year is a next year, I'm gonna clear my plate off of a lot of things. And then we'll see what I can build.

P Paul Cutler 11:11

On That's great to hear. So between the sensor, watch the LCD FeatherWing. And the open book, a lot of your projects are battery powered, how do you manage power consumption to make them last as long as possible before needing to be charged?

J Joey Castillo 11:23

Yeah, so this is kind of a pretty relentless focus of mine, there's two pieces to it, which sort of boil down to like the phases of like standby mode, and deep sleep mode. So pretty much all the microcontrollers, you're going to work with in circuit, Python, Arduino or anything. They've got two modes, they've got an idle mode where the CPU is running, and they've got a standby mode where the CPU isn't running. So the more time you can spend in that standby mode, the better better off you are. It's like with the LCD, when, for example, I have a clock demo in Arduino, and it blinks the colon twice a second. So just like you know, blink, blink, blink blink, like a ticking clock. So you could do that by blinking the colon and then calling the delay function with a you know, 500 milliseconds, and then you delete, toggle the call and again, you know, so that would work. But if you call delay, then you see he was running because it's using the system tick to determine that amount of time. But he uses Adafruit sleepy dog library to sleep for 500 milliseconds, then the CPU can actually go to sleep and into standby rather. So that's like 99.9, something percent of your second that is just not even awake. It only needs to be awake for that fraction of a second that it's bursting some data out over the I squared C bus. So that's kind of one part of garbage, just trying to stay in standby mode as much as you can. In the case, the LCD feathering also did a kind of a cool trick with the LCD signal routing, where you only have to send one byte over the bus to blink colon are the indicators. And that's another kind of fun trick, the less data you have to send out over I squared C or SPI bus, like the less time your CPU is awake number one, and number two, especially with I squared C, there's fewer zeros on the I squared C bus, which means fewer less current kind of flowing through those pull up resistors. So that's kind of in the weeds. But that's another one where it's like yeah, the less you can do the better. So the other thing, and this is still good enough to stand by Mike is this applies to both. The other thing is just being really careful about what other gadgets in your gadget are getting powered when you're in sleep mode. So with like some early open book prototypes, this is the one that I kind of, I kind of threw the kitchen sink at the early open book prototypes that had a NeoPixel it had a flash chip, it had a it had a microphone preamplifier to do voice commands out of a headset, it was over the top, but I was looking at it and I was like, why is it consuming so much power? It's not supposed to like my

processor is not supposed to consume this much. And then I realized every gadget that had hung on the Christmas tree was drawing power. My mic preamp was drawing power, my NeoPixel was drawing power. So kind of everything on your in your gadget, like if you tie it to, you know, VCC, or three volts or whatever, it's going to be drawing its quiescent current, all the time. So now on the new kind of open book boards, I gave all the power to those peripherals behind a MOSFET power switch. So let's say that you wanted to make a pigeon camera like you know, put next to a bird's nest or something like take photos of that is actually a project they've built at one point, it took photos of the pigeons, land a time lapse. At first, I just powered up one of these little serial cameras, and the battery died within a few hours, because a few hours, but a few days because it was powered all the time. Whereas if you only turn on that camera, when you are ready to take a picture, and then you turn it off afterwards, then you're going to get a lot more, you're getting a lot more life out of that battery. So that's one thing. The other interesting one. So one of the early open book prototypes, I get it all my power behind that MOSFET power switch. But even when I turned off power to all those peripherals, it was still drawing more current than I was expecting to so I'm like, What is going on here? And what I realized was some of the signal lines that actually went out to the paper driver and stuff were being held high I even though I was going into this deep sleep mode, so it was back powering some of these things that even though their power was turned off, the signal lines were a microcontroller, were back powering the gadget. So that was causing them to draw more power than I wanted them to. So I had to drive those signal lines low before I went into deep sleep to make sure there was no current whatsoever flowing to them. And that was another way to get a little more battery life because you just to be really careful and make sure that everything that can draw current is configured so that it can't drop current, when you're in that kind of mode, where you want to be asleep or drawing what's current. So the other, that's, that's half the equation, standby stuff. The other big side of thing is deep sleep mode. And this is especially interesting to circuit Python, folks, because, well, first of all, not all chips have this, but especially like the ESP 32 series and the feather and four series, they've got a few pins that can do pin alarms, and they've got a real time clock that lets them do alarms. So with a pin alarm or an alarm, you can put your circuit python script into deep sleep mode. And then it goes into this like really, really low current consumption mode where it's on the order of like, single digit to low double digit micro amperes. And it can stay there like, you know, it can stay there indefinitely until a pin gets pressed, or it can stay there for you know, seconds, minutes hours, until you're ready to wake up. And it does restart in the beginning of your your script. But in a lot of cases, that's fine, because like, let's say hypothetically, you wanted to do that clock example from earlier. But let's get rid of the blinking call. And let's just like update the time, once a minute. So you could update the display and then enter deep sleep mode for 60 seconds. And then at the end of 60 seconds, your script starts over again, it updates the display with the value of the real time clock, which is the current time. And then at the end of that just go back into deep sleep. So now you're spending like 99.9% of a minute in deep sleep mode. And that's where you can get into like real real like low power savings. So actually, I did this kind of as a stress test. It was a weather station type gadget I made using the ESP 32 s two and the LCD FeatherWing. And one of those 400 milliampere hour batteries that fits between the federal headers. So really small, we're talking like one or two cubic inches. So small gadget, and the setup was that it would turn on or sorry, it would wake up, it would turn on the temperature sensor, it would take a reading it would turn it off, then it would turn the Wi Fi on upload that data at Adafruit IO, and then go back into deep sleep for 15 minutes. So it only had to be awake for maybe like 1015 seconds every 15 minutes. So you're up for a minute an hour, that set up lasted two full weeks on that little 400 milli ampere milliampere hour battery before I needed to recharge it. And if I wanted to make a more like robust weather station that I could leave out for a month, double the battery 100 milliampere hours and you'd get like a month of battery life. So yeah, like deep sleep ends up being a really powerful tool for making circuit

Python scripts, low power to the point where you could actually write a circuit Python gadget, deploy it out in the world on a solar panel and a battery and expect it to last for quite a while just on the sleeve.

P **Paul Cutler** 18:05

That's really impressive. Last question that I asked each guest you're about to start a new project, which microcontroller do you reach for?

J **Joey Castillo** 18:13

So I gotta say it's a it's not the new hotness, it's kind of an oldie but goodie, the Feather M zero basic. It's first of all, I'm all in on the feather ecosystem, just because I make battery powered things and the idea of that battery powered gadget, you just plug into USB to recharge. That's, that's huge. That's like the framework for making all kinds of cool stuff. And then the Sandy 21 is just kind of like the Swiss Army knife of microcontrollers. It does so many different things and it does them all well. So I'm probably going to find whatever I'm about to do will fit well with what the Sandy 21 needs or has available to me. And then the classic one in particular because two things. One, I mentioned about low power stuff, there's no NeoPixel no flash chips, so there's very little in the way of extras, so not a lot can like draw extra power for me, but also the little prototyping area, it's really useful to make it make the gadget into whatever you need it to be. So like, I'm building a solar power gadget for a friend who is doing an art installation. And then we'll prototyping area becomes a direction pad for navigating a series of menus. Or there's a feather m zero running my LCD FeatherWing testing setup. And that prototyping area gets gets a power switch just to turn it on and off so I can like leave it off most of the time. If you want to make a atmospheric sensor like solder in one of those little simple humidity sensors and little prototyping area. You don't need to dedicate a whole protoboard when you just like turn the Feather M zero into whatever gadget you need it to be. I think that's a huge usability win. So yeah, if I'm reaching for if you told me nothing other than I'm reaching for board to do a new project, it's probably going to be the Feather M zero basic.

P **Paul Cutler** 19:47

And if people want to learn more about your products, where should they go?

J **Joey Castillo** 19:50

So at this point, I'm still on Twitter. We'll see how that goes in the coming weeks. Otherwise, I've got a count on Mastodon which is LinkedIn, my Twitter handle it and, oddly specific objects.com is the website where you can check out the things I've made. And there's actually, there's actually a link to shop that oddly specific objects.com to buy some of the stuff where you can buy the federal mainland Adafruit. So

P **Paul Cutler** 20:13

that's great, Joey, thanks so much for being on the show

that's great, Joey, thanks so much for being on the show.

J Joey Castillo 20:15

Thanks for having me. Really great conversation and I appreciate it.

P Paul Cutler 20:22

Thanks to Joey for being on the show and thank you to PCBWay for their sponsorship of the circuit Python Show. For show notes, transcripts and support the show is at Circuit Python show.com. Until next episode, stay positive