**“What was your background before landing at IU”**

COOP at intel. Coop isn’t glamorous but he had access to people doing interesting things in the embedded space. Supporting delco and other big consumers of intel chips for embedded systems. AD196 MCU, was used by cumins as well as delco, which was how Bryce met his next boss who gave him a chance to interview at cumins after his COOP.

Started forCumins @ cumins electronics (cumins subsidiary). Took fuel system specs for the electronics and build the [brain] box. System that manages fuel flow to the cylinders. Product design engineer: hardware design. wrote software to the extent of “bringing up the system”, have to be able to write software to ensure that the hardware works the way you designed it, but that software would never see production. Write code that would run on the module when it would be tested in production: these tests were heat cycles that the systems were ran through to ensure that solder joints weren’t breaking during the heat cycles. Final test: ensure different inputs, simulate vehicle speed and make sure module measured it correctly, fire an injector and make sure the current wave form looked correct. Most of the job was assisting in the writing of the spec, designing the hardware, implementing board through all of the stages of production, and delivering a module to the main engine company.

Joined an engine development team where he was responsible for the module, the harness, the sensors. The whole thing that went into the whole engine program.

(DEVELOPMENT WORK)Came to IU to be a designer, they needed engineers to help design systems that are used in C335(student projects), research. FOR 10 years

(needed someone to lead merger between IT teams for INFO and CS) –

^during this time^ Was still doing research for modes of analog computation, low power neural networks.

(Started a startup) – patented some ideas and got some seed capital, enough to run the company for two years. (still worked half time for the university, teaching 335)

(came back to university to teach)

**“In your time of dev, you’ve seen the advent of IOT and revolution of embedded systems, what was it like when it started? Where do you see it going now that we have a lot more prevalent wireless technologies”**

THEN: When I started it was before the advent of Flash memory, so they had to burn EPROMS, which expose UV light by putting it in a little thing that looks like an oven. EPROMs severely limit the kinds of systems you could develop. ECU world had flash, but it was externally exposed where all of your memory wasn’t on the Die, so it had to be mapped to the IC.

NOW: we have processors and communication peripherals that dwarf what used to be used when it comes to power and [re]usability. I look to see low power communication protocols being integrated more right now SPI and I2C, ethernet is becoming more commonly integrated (ex: Rpi), he expects wireless transceivers to be integrated and it will become cheaper and lower power to do these designs. He hopes that a network that is IOT available grows to become as prevalent and as integrated into our infrastructure as Cellular is today. We could be anywhere in the everyday world and expect to be able to send a text, make a call, or query google. He finds this new infrastructure important because Cellular is not feasible for IOT because its use is predicated on a financial relationship with the provider. It’s intractable to have 1000 sim cards for each of your sensors in the field. He gives SIGFOX in France as an example of this desire. It is a low bandwidth IOT specific network that’s subsumed into the infrastructure similar to ethernet or power. It’s subsidized and at some level someone is footing the bill for it, but it’s not a direct financial relationship with the carrier.

Why is it important? It will enable thousands of autonomous sensors, this will eliminate IoT being tied to your wifi.

**“What words of wisdom do you have for the person who is just getting into the hardware game.”**

Join one of the learning communities whether it be Arduino or mbed, etc. Where there are many people with a lot of experience. Because the cost of these development kits have become very affordable. The impetus to buy a few hundred dollar development kit is out of the way. Using Arduino or mbed, you can compile, download, debug for free. I would start in these ecosystems and leverage this community as much as I can to both learn how to program and to build some real systems. These communities and platforms are so powerful and have so much open source knowledge that you can build a real system almost immediately. That is where you are going to learn the most. Try it. Try to build it. Do it; you’ll run into all kinds of problems, and it’s those problems where you’ll actually learn. Being a beginner developing hardware and you’re a new learner the worst case situation is where you are in a custom platform where you cannot separate out whether the issue is your hardware that you build, or is it the software that you wrote. The ambiguity between those two problems require a lot of experience to work through.

That is a major reason why it is better to be on one of these popular platforms because the probability that someone has had your problem or a similar one is high. That is one of the powers of leveraging these platforms is that you will probably be able to figure out what you’re doing wrong on a forum, and if not there will be someone with the experience you’re lacking to help you diagnose your mistakes.

**“You’ve seen the startup realm. You and I have talked a lot about the value proposition and how that is above all the most important part of the startup game (particularly when it comes to getting funded). What other skills or information is critical for people eager to join this domain?”**

Minimum viable prototype: enough of a model of your system to probe the entire design constraint space is essential. Because often times what you’ll end up doing is you’ll get way down the road on some assumptions, and if one of those assumptions turns out not to be true it changes a lot about your system. Many times you could have identified this problem or bad assumption very early with a minimum viable prototype, and gone another direction. Then you will have the opportunity to invest your design and development time whether it be software or hardware elsewhere. Design and development time is precious in the startup world because you are almost always constrained by resources. Additionally don’t be afraid creating a startup because it’s going to fail, and also don’t be afraid of starting another startup because you’ve had one fail. You will learn many lessons from the process irrespective of the ultimate outcome.