Project Closeout

Group #33 | Power Management for Collective Ground Robots | Assignment Description

Eric Prather prathere@oregonstate.edu

Kyle Noble nobleky@oregonstate.edu

Miguel Ruiz ruizmig@oregonstate.edu

Thomas Snyder snyderth@oregonstate.edu

Table of Contents

Table of Contents **2**

Design Impact Statement **3**
- Public Health, Safety and Welfare Impacts 3
- Cultural and Social Impacts 3
- Environmental Impacts 3
- Economic Factors 3

Project Timeline **5**

Scope and Engineering Requirement Summary **6**
- Adaptive Power Management 6
- Charger Safety 6
- Charging Algorithm 6
- Data Reporting 7
- Graphical Interface 7
- Hardware Footprint 7
- Server Design 8
- Wireless Latency 8

Risk Register **9**

Future Recommendations **10**
- Charging Pad Design 10
  - Recommendation 11
- Robot Software Design 11
- Swarm Server 12
- Sensor Processing Board Auto-detect Sensors 12
  - Recommendation 13
- Sensor Processing Board Logging API 13
  - Recommendation 13
- Sensor Processing Board Charging Notifications 13
  - Recommendation 13
- Sensor Processing Board Sensor Priority 13
  - Recommendation 14
- Sensor Processing Board Crazyflie Expansion Port Sensors 14
  - Recommendation 14

Works Cited **14**

Design Impact Statement

This design impact assessment is intended for audiences interested in the ethical decision making and counter measures to address ethical problems in the power management for collective robots ECE capstone project. The assessment will consider four impact areas: public health and safety impacts, cultural and social impacts, environmental impacts, and economic impacts.

Public Health, Safety and Welfare Impacts

The HMTLab publishes papers to the general academic community, and the findings within the lab ultimately progress the field towards real-world implementations of swarms. These may be used for the benefit of humanity, but are often also weaponized [1]. The HMTLab participates in OFFensive Swarm-Enabled Tactics (OFFSET), one of DARPA's swarm research projects. As the lab's activities are explicitly designed to be conducive to the military, any continued support or operation of the lab may in turn be unethical. We psychologically distance ourselves from this moral culpability with the rationale that this team was tasked with creating a power management system, not making a weaponized robot. Besides, there are other positive areas that swarms can assist in, such as search and rescue, surveillance, cordoningan area, and more [2]. We furthermore substantiate that the quality of our work cannot be sacrificed, as when swarms are deployed over populated areas, failure becomes increasingly dangerous.

Cultural and Social Impacts

Ethical considerations encompass code development for graduate and undergraduate students involved in the lab. For instance, where there are cameras, then the audiences that observe these robots will be captured and stored on a server. This data may be used for individual identification; however, these processes are not within the scope of this project because the project partner does not require sensor selection, so we once more psychologically distance ourselves from culpability. Besides, using visual sensing on robots improves spatial awareness and reliability in cluttered environments. Additionally, visual sensing enables facial recognition and surveillance for public safety at the cost of individual privacy. The Association for Computing Machinery (ACM)'s Software Engineering Code of Ethics expresses this value in stating that data must be accurately collected, with collection techniques adhering to the law [3]. The lab using the team's software must adhere by law both to this and its own standard to mitigate this risk, and as such we forward the culpability to them.

Environmental Impacts

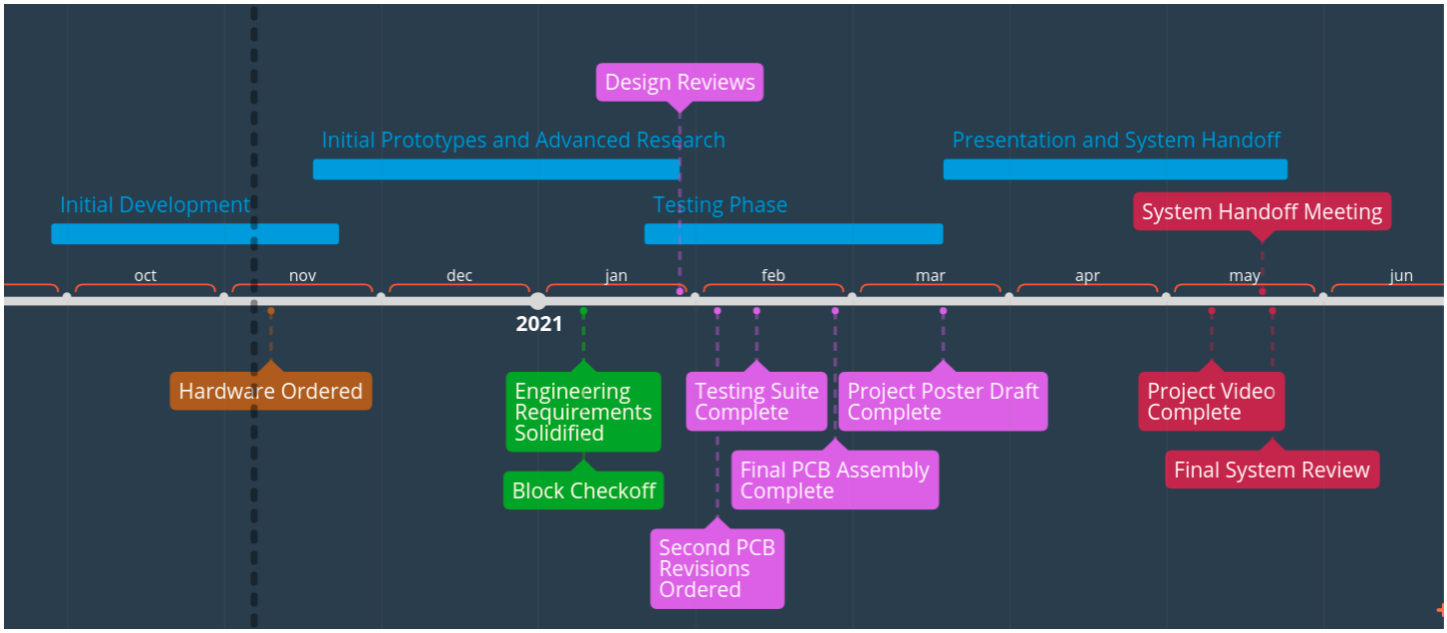
The hardware in the system directly affects the project stakeholders and is difficult to change, a fact we utilize to absolve ourselves of culpability. Nevertheless, it is important that all ethical considerations of the hardware be expressed and addressed in early development rather than at the time of deployment. An example of a document to support this process would be the RoHS standards on rare-earth and toxic materials [4].

Economic Factors

As swarm technology becomes more prevalent, the production will grow, the price per robot will decrease, and the new industry may provide unskilled labor jobs to the benefit of the economy. The price of generic robot parts will also decrease, making robotics more accessible to the a wider variety of ventures, including

education, hobby, and research. However, the consequent increase in resource usage - especially with the current COVID chip shortage - may exacerbate supply chain scarcity.

Project Timeline



Scope and Engineering Requirement Summary

The project requirements and scope are principally defined by the source material in the client outline of the project, authored by Jenifer Leaf. Following is the extrapolation of specific ECE 44X compliant engineering requirements. The project partner requirements are specified in a subsequent table.
These requirements

Adaptive Power Management

PPR: Recommend whether some or all of the robot's functionality needs to be turned off while the battery is recharging.
ER: The system will inform the robot (external) which sub systems need to be shutdown until the battery is charging via the Qi sub-system.
Evaluation method:
1. Set a robot, with all peripherals on in front of a charging pad
2. Draw a load from the robot that will cause the battery to discharge while charging from the wireless system
3. The system will send a message to the server indicating what is being shut down.

Passing criteria:
The robot sends a status update indicating that it has turned off some part of the robot and the battery registers as charging. The charging light on the robot is lit.

Charger Safety

PPR: The base charging station solution must be safe. It must not catch the testbed on fire.
ER: The system will charge a robot without the peak temperature exceeding 425 degrees fahrenheit and peak power consumption staying below 2400 Watts.
Evaluation method:
1. Connect the input power of the system to a DC power supply.
2. Plug a discharged battery into a robot.
3. Align the robot with the primary coils of the charging station.
4. Turn on the charging base station.
5. Monitor the voltage and current output of the power supply and measure the hottest part of the circuit for a duration of 10 minutes.
6. Calculate the peak power by multiplying peak current and voltage
7. Measure the voltage of the robot's battery.

Passing criteria: Hottest temperature does not exceed 425 degrees F (combustion temperature of MDF), and peak power does not exceed 2400 Watts. The voltage of the robot's battery after charging is greater than the initial voltage.

Charging Algorithm

PPR: The robots must be able to predict when to recharge their batteries.
ER: The system will ensure that 5 robots will operate for twice the expected life of their batteries (as computed by the battery’s amp hours and robot average power draw) while performing tasks between each charge without user intervention.
Evaluation method:
1. Measure the power draw of a robot driving in a circle for 20 minutes (1200 seconds).

2. Compute the expected battery life based on the equation: hours of operation = (battery amp-hours) / (average current)
3. Run 5 robots with the system. Charge the batteries beforehand
4. Make the robots drive for twice the nominal battery's operational amp hours as predicted by the previous calculation
5. Count the number of robots that are out of battery
6. Check voltages on the batteries
Passing criteria: After the test, all of the robots have recharged at least once and are still able to move without human intervention.

Data Reporting

PPR: Identify an appropriate protocol for exchanging messages between a PC and the UGVs. Define the API, and implement the server portions of the API on the PC. Design how data will be logged during experiments, and implement data logging.
ER: The system will expose data from the robots collected during experiments via a single system endpoint. The data must include, for each robot, at least once but no more than once every half-second of the experiment: robot state, battery voltage, current, power, battery temperature, time since power on, and a customizable field.
Evaluation method:
1. Release a fully charged robot to operate.
2. Log the system output data for exactly 10 seconds.
3. In the log of data, count the number of robot state, battery voltage, current, power, battery temperature, time since power on, and a customizable field present.
Passing criteria: If the number of elements counted is between 1 and 20, this engineering requirement has been met.

Graphical Interface

PPR: The wireless monitoring interface must be scalable to hundreds of robots.
ER: The system must update the user every 1 second when receiving data from 100 robots (real or simulated).
Evaluation method:
1. Program 100 real or simulated robots to send different position data every 0.5 seconds
2. Track each robot update with the server interface. Make sure each update has a timestamp when stored in the database.
3. Average all the time differences between the update and its previous update.
4. The average time gap between two consecutive updates must not exceed 1 second.
Passing criteria:
The average time gap between two consecutive updates must not exceed 1 second.

Hardware Footprint

PPR: Any additional hardware should be minimal, not adding to the footprint of the robot by a substantial amount.
ER: The system will not extend past the current cubic dimensions of the robot (73 x 64 x 59 cubic mm width, length, height) by more than 1 cm.
Evaluation method:
1. Fit the system to the robot

2. Measure along each dimension (length, width, height) of the robot
3. In every dimension, the robot outfitted with the system is less than or equal to 1 cm larger than the original robot.
Passing criteria: In every dimension, the robot outfitted with the system is less than or equal to 1 cm larger than the original robot.

Server Design

PPR: The server must run on Ubuntu Linux and be supported by a well-documented API.
ER: The system will expose an API accessible to external users. All API endpoints will be comprehensively documented in terms of all possible parameters and data schemas using an OpenAPI compliant documentation framework. Three endpoints will be described: POST rb/log: Adds an arbitrary log to the server. Message body contains any JSON object, which will be stored as-is; POST rb/status: Describes the current state of the robot for data collection. Message body contains JSON containing sensor data; GET rb/command/{MAC}: Gets the current orders issued to the robot with address MAC. Response body contains JSON describing the robot's present command.
Evaluation method:
1. Initialize the runtime with an empty history and populate it with an arbitrary dataset.
2. Prepare a list of all API calls exposed by the server. Record a cross-factorial testing matrix of each of the parameters to the call.
3. For each call, evaluate the cross-factorial matrix and compare the results to the expected outputs based on the arbitrary dataset passed in. This may require observing the state of the server.
Passing criteria: All three API call outputs the expected values as calculated independently by the testing suite.

Wireless Latency

PPR: The wireless monitoring interface must be scalable to hundreds of robots.
ER: The system will maintain average network latency of less than 300 ms on API requests while 10 robots are sending status updates to the system.
Evaluation method:
1. Place all 10 robots equipped with the system on the testbed. Turn them on.
2. Over a period of 5 minutes, record packet latency every time a packet is received.
3. Average the latency over all packets.
Passing criteria: The average latency is less than or equal to 300 milliseconds.

Risk Register

Table 1: Risk register, last updated April 30

Priority	Risk ID	Risk Description	Risk Category	Risk Probability	Risk impact	Performance Indicator	Responsible Party	Action Plan
1.8	**T11**	COVID exposure	Timeline	60%	High	Individual tests positive for COVID.	All except the person who is sick	Transfer
1.65	**T8**	Parts on BOM go out of stock or are delayed due to pandemic.	Timeline	55%	High	Stock is low, shipping times high.	Kyle or Thomas	Reduce
1.5	**E3**	Network protocol does not meet latency and traffic management requirements	Technical	50%	High	The RESTful service is not able to meet the project's rigorous technical specifications	Thomas Snyder	Avoid
1.5	**E5**	Robot locomotion dependency violation	Organizati on	75%	Medium	The robot is not able to navigate the testbed effectively enough to reach the general vicinity of the charging station	HMTLab	Transfer
1.2	**E1**	Centurion UGV has a fundamental shift in capability due to its engineering being ongoing and activity both inside and outside of the development team	Organizati on	60%	Medium	Robots cannot enact battery and communication algorithms	Jennifer Leaf	Retain
1	**T3**	Lack of Machine Learning Resources	Technical	50%	Medium	One hour spent without finding an easily understandable tutorial	Eric	Transfer
0.9	**M5**	server side communication failures	Technical	30%	High	the server fails to communicate with all of the agents due to scalability problems.	Miguel, Eric	Retain
0.75	**T2**	Qi Standard Self-destructive Interference	Technical	25%	High	Charging coil does not output a voltage, or outputs an unexpected voltage, when placed in the proper	Kyle or Thomas	Avoid

						arrangement for charging		
0.75	**T7**	Wireless Infrastructure at Mill Race cannot service the number of robots that are needed to test the system	Technical	25%	High	Robots are unable to connect to wifi or connect and then cannot check in with the server.	Miguel or Eric	Avoid
0.75	**E4**	Machine learning is not able to effectively model the lithium ion batteries	Technical	75%	Low	The machine learning technical demo demonstrates the technology to be inferior to competing models	Eric Prather	Avoid
0.7	**E8**	Tensor field predictive models inappropriate through external analysis by other HMTLab activities	Technical	70%	Low	Test on data state fetch from server results in insufficient data for analysis as evaluated by project partner	Eric Prather	Retain

Future Recommendations

Our project is partitioned into several areas which will require attention from different expertise areas within the HMTLab. These areas are in the domains of the charging hardware design, robot software design, and server software design. The hardware on the robot is of sufficient quality such that no immediate future recommendations are present. We conclude with some exciting anticipated directions for future development on the Centurion robots' sensor processing boards.

Charging Pad Design

Several supply-chain and form-factor risks hindered the completion of the wireless charging docks used by the collective robotics testbed. Our original requirements specified that the charging pad system would be able to charge 10 robots concurrently. A prototype board was designed and tested to charge a single robot. This prototype has switches which control a number of settings on the wireless transmitter IC, one of which includes the number of coils that the charging pad will use. Using multiple coils enables free-positioning, meaning that when a receiving coil is aligned with an array of coils, the transmitter IC will determine which coil should activate in order to send power to the receiver. However, the majority of our testing used the single coil mode on the transmitter IC. The prototype board can theoretically support up to four coils, but the multi-coil setups require further testing. Once the functionality of the multi-coil modes can be verified, the prototype board could be simplified into a cheaper and smaller board. This can be done by removing the switches and tying the function pins to either the appropriate power rail or ground depending on the desired settings. This will remove all the switches (except maybe the power switch), which is a significant expense when ordering these boards in mass quantities, and this will also allow the PCB to be smaller, which means cheaper PCB manufacturing.

Another problem that was raised is that the charging hub needs to be able to support 2 charging pads on each of the four sides. Each side is about 20 cm wide, meaning that each coil array should be less than 10 cm wide. This means that at most 3 charging coils can be used per charging pad in the hub. When 3 coils are arrayed according to the Qi specifications reference power transmitter design A6, the coil array will be around 9.5 cm wide, which will allow two sets of charging coils to be placed side-by-side on each face of the hub.
One feature which was requested for the charging pad design was to be able to reuse the coils from the broken off-the-shelf charging pads. These pads, however, used coils that did not have the correct properties (i.e. inner/outer dimensions and number of turns) for use in an A6 transmitter design. None of the Qi wireless transmitter reference designs with free-positioning use coils with the same properties that the coils from the broken charging pads have. This indicates that using these coils in a free positioning transmitter design would be extremely challenging, as one would need to design the system from scratch using the Qi specifications. The solution that we employed to address this problem was to unravel the coils, solder 2 together and try to shape the wire into the correct dimensions and number of turns as specified in the A6 transmitter design, using tape to hold the shape. This is not an ideal solution as it requires a lot of manual labor and the coils will not be accurately constructed when shaped by hand. These coils also will not have the ferrite shielding that is typically recommended for charging pad coils to be mounted on. However, this shielding is usually only needed when the coils are very close in proximity to the charging PCB.

An additional challenge for the charging pad is that we were unable to acquire the hardware needed to create 10 charging pads due to supply-chain risks, so the requirement to charge 10 robots concurrently was changed to only charge one while the other 9 outputs are connected to high-power potentiometers each with a resistance of 42.8 Ohms, since the nominal current load of a single charging board is about 280 mA and the supply voltage is 12 V. This will effectively simulate the power load that the other charging boards would put on the system, while verifying that the actual charging pad is still functioning.

The final challenge for the charging pad is that the charging pad failed to charge the robots. Although the charging pads worked when charging a phone, the transmitter IC entered an error state after a few seconds of charging the robot. The types of errors that were specified in the datasheet are over-temperature, over-current, and short-circuit protection. We attempted to troubleshoot the error, but were unable to determine which of the errors was causing the problem.

Recommendation

To address each of these issues, we recommend the following: First, we recommend verifying the functionality of the prototype charging PCB in 3-coil mode and finding which pin function configurations are ideal. The next step is to refactor the prototype board into a simpler design by replacing the switches with direct connections to the desired inputs on the PCB. This will allow for a smaller hardware footprint and fewer parts to order when ordering these boards in bulk. Next, we recommend purchasing Qi reference transmitter design A6 compliant 3-coil coil arrays for each charging board since reshaping the existing charging coils is not a very efficient or effective solution in the long term and the existing coils were not designed to be used in a free-positioning transmitter design. The only way that the coils could feasibly be reused for this project would be if they were used in a single coil transmitter design, which would require the robots to be able to accurately back up to a single transmitter coil. To reduce the number of errors to troubleshoot for, it may be worth removing the over-temperature protection and instead connecting the TH1, TH2, and TH3 pins to the 3.3V power rail. The over-current protection can be made less likely to throw an error by using a smaller resistor (e.g. 25 mOhms) for R4. The short-circuit detection can be investigated by probing the SCDETn pins on an oscilloscope and checking if the time until oscillation stops exceeds 100 microseconds.

Robot Software Design

This capstone project was able to successfully meet its requirements, although several design concessions were made on the design of the robot in order to meet the rapid prototyping requirements of the demonstrations. Following the capstone project, when resources become more readily available, there are several areas in which the design of the robot software could be improved.
First and foremost, the automatic charging and wireless communication functionality would do well to be removed from the application layer and introduced to the library layer. While our proof of concept application does well to demonstrate the system's flexibility and capabilities, some additional work could be done to ensure the scalability and abstraction of the present solution.
Furthermore, the software infrastructure used to design applications for the robot is ill-supported by a robust deployment protocol. Packages are managed manually and environment variables must be set by modifying source files therein. Building and deployment are inseparable through the provided protocol, and several features at the library level such as Arduino OTA flashing are unutilized. The library layer furthermore exposes a variety of compile-time warnings and runtime failures which must be manually discovered and accounted for at runtime; for example, the serial output stream is cluttered with I2C debug messages indicating failure even when the I2C system is operating nominally. Many of the key GRITSbot library files seem to have been implemented at the amature level and contain a variety of design anomalies and inappropriate abstractions, particularly troublesome in the case of wireless communication. Augmenting this process with relevant build scripts, an environment variable management system, and package management protocols may be appropriate. In addition to project-level package management, some mechanism for IDE synchronization may be appropriate so that VSCode does not need to be configured with the precise required dependencies on each repository instance. Revision to the library layer is needed. The capstone team has discussed moving away from the arduino IDE altogether if possible.

Swarm Server

The swarm-server's core architecture is robust and scalable; adding new endpoints and data models for the database should take a matter of minutes in most cases and can be patched in a backwards-compatible way when necessary. A summative developer manual may be useful in addition to specific documentation, highlighting key required knowledge for efficient work on the project including that of automatic OpenAPI documentation generation and object-relational mapping.

Many features extraneous to the core requirements of the capstone project were started and rudimentarily validated, but not included in the final system. Of primary interest are the long-term storage, rasterization, and predictive algorithm systems. All of these, especially the first, have a variety of use-cases and would do well to be finished in a timely manner.

Several features described for capstone are entirely useless outside of the context of the highly specific system integration testing requirements and should be deleted. These are identified in the <remarks/>.

The key limitation of the server design is in the difficulty of deployment; although all of the dependencies are clearly tracked and managed, in order to maximize the platform's usability, we recommend **containerizing** the application alongside all necessary runtimes (e.g. a SQL database provider). We propose the best way to do this to be with Docker.

Lastly, a variety of <remarks/> have been left throughout the project detailing minor necessary work items. In addition to following through on these, it would also be appropriate to switch to a proper issue tracking system for more organized collaborative future development.

Sensor Processing Board Auto-detect Sensors

The system's sensor processing board currently contains a lightweight sensor API. The sensor API allows the developer to communicate with any sensor on the unified sensor I^2C bus. During processor initialization, the sensor API loads a driver for each sensor, but this process is coded by the developer. The sensors were not a project requirement, but incidental in creating the sensor processing board, so a bare-bones groundwork was laid that is functional. Making a rich sensor API only requires software development.

Recommendation

Moving forward, assign unique I^2C addresses to each sensor. The unique address can be used as an identifier, so that the sensor processing board can use an initialization stage that reads the addresses that are connected to the bus. Initialize every sensor in the sensor initialization function, then add a second step in initialization, which maintains an array of all possible sensors and uses the addresses to check whether the sensor is active. When the sensor is active, move it to an "active sensor" array so that the sensor module only uses the drivers that are currently connected.

Sensor Processing Board Logging API

The system currently exposes a logging API for internal logging to an onboard SD card. SD card logging was the only requirement for the logging module; however, the current logging API is extensible to other logging destinations.

Recommendation

The logging API currently has hooks and examples for adding extra logging destinations. These can be utilized as future needs arise. Some examples in the code include logging via a UART bus and logging to the mainboard via the I^2C commlink. Adding a new destination requires entries in a few different areas of the code, however all functionality can be implemented in "modules/log.c" and "modules/log.h".

Sensor Processing Board Charging Notifications

The sensor processing board is connected to the Qi Wireless Charging deck's n_charge_sense pin. This pin goes low every time the system is charging. This feature may be used to verify the robot charge state. For example, if the sensor processing board senses charging, but the mainboard does not sense that the battery is charging, then the sensor processing board may shut down sensors based on a level of importance.

Recommendation

The work to implement the charging notification will span the sensor processing board and the mainboard. The charge sense API is in place. The only modifications that are needed come with respect to the communication link and the implementation in the mainboard firmware. The message type already exists. The mainboard simply needs to issue a request and ask for a response over the I^2C bus.

Sensor Processing Board Sensor Priority

The main focus of the *Power Management for Collective Robots* capstone project was to allow the robots to charge and intelligently determine their charging state on the fly. The sensor processing board has the ability to shut down sensors and “adaptively manage power” by shutting down extraneous and power hungry sensors while the robot is charging. Assigning priorities to the sensors gives information so that the sensor processing board can intelligently shut down sensors.

Recommendation

This recommendation may be implemented alongside the “Auto-detect Sensors” recommendation. The sensor priority will require adding a new data field in the struct, probably an enumeration. The enumeration should contain at least three entries ranging from “critical” to “safe,” or some labels that indicate the sensor’s importance to the system. Depending on the charging state, the sensor processing board will be able to shut down the sensors in tiers. For example, suppose the levels are “critical,” “semi-critical,” and “safe” and the robot is at a charging pad, but not charging. The sensor processing board may then shut down all “safe” sensors to allow the robot to charge. Next, the “semi-critical” sensors may be shut down if the mainboard does not detect charging still. This adaptive power management will enable decreased charging times by decreasing the Centurion’s idle power consumption.

Sensor Processing Board Crazyflie Expansion Port Sensors

The addition of the Crazyflie expansion port (CEP) enables the possibility of using sensors developed by Bitcraze AB. The communication interfaces that are exposed in the CEP have been tested to ensure functionality; however, there have not been any abstraction or attempts to create a sensor API that interacts with the CEP.

Recommendation

The CEP is currently used solely by the *Eminus* localization sensor. The maximum number of sensors that will occupy the bus is two, given that the wireless charging pad is paramount to long-term experiments. Thus, future work should implement sensor interfaces ad-hoc because the number of sensors on the CEP is low and the chances that they will change is also low due to the port’s proprietary form factor.

Works Cited

[1] Offset envisions swarm capabilities for small urban ground units, Dec 2016.

[2] Melanie Schranz, Martina Umlauft, Micha Sende, and Wilfried Elmenreich. Swarm Robotic Behaviors and Current Applications. Frontiers in Robotics and AI, 7:36, apr 2020.

[3] Software engineering code. https://ethics.acm.org/code-of-ethics/software-engineering-code/, Dec 2018.

[4] Rohs compliance faq. https://www.rohsguide.com/rohs-faq.htm, Nov 2020.