UB-0026 Engineering Intramural

SnowBot

Report Title

Sensor Acquisition and Integration for Automation

Key Stakeholders:

Project Sponsors:

- **Bob Girardi, formerly CEO of Softek,** Computer Science BSc. 80’
- **UB Sustainable Manufacturing and Advanced Robotic Technologies** (SMART COE)

Project Team:

- **Christopher Perez**, Computer Science BSc. 20’
- **Benedict Cutri,** Mechanical Engineering BSc. 21’
- **Tamaghan Maurya**, Computer Science BSc. 21’
- **Kevin Johnston,** Electrical Engineering BSc. 21’
- **Randy Chung,** Electrical Engineering BSc. 21’

Abstract:

In response to the ever-growing size of the technical literature associated with machine learning and object classification, increasing amounts of focus, whether it be in academia or in industry, have been directed to the possible application of these statistical learning methods and optimal control algorithms in automating systems across all forms of industry. At the center of this interest lies the question that has been in the zeitgeist of the academic community for the past few decades, namely: **How, if possible, could we teach machines to make judgement calls like we do? What could its implications be?**
Although we might not be able to answer this question right now, understanding the challenges associated with the development and implementation of these methods to traditional electro-mechanical systems like a snow-blower could serve as a crucial first step to realizing a future where systems could be designed with broader design goals in mind and yet fulfill an expansive set of roles and inch us forward to understanding how we might answer this question.

Introduction:

At the core of this project was the structured exploration of the technical literature associated with machine learning and possible configurations of available hardware to implement these techniques in practice and achieve an autonomy level between **Level 3** and **Level 4** as defined by the five-tiered classification of autonomous control introduced by the Society of Automotive Engineers. The primary aim of this project is to leverage this technique to allow SnowBot to carry out its responsibilities with no human input **in the context of a controlled environment**.

With the conclusion of this project, the following objectives were consistently met:

- Algorithms that allowed SnowBot to identify and classify objects in its environment were identified and decision matrices used to choose optimal algorithms
- Hardware requirements for algorithms were defined and the optimal configuration of sensors that met these requirements were selected and documented
- Project objectives, customer and engineering requirements, milestones, schedules and risks culminated in the creation of the project charter for the project.

- Regular weekly meetings were conducted between the project team and Dr. Olewnik, who acted as a proxy for the stakeholders, namely SMART COE and Bob Girardi, for the duration of the project.

In terms of scope, this phase of the project extends to the research and identification of relevant technologies associated with the implementation of an autonomous vehicle, and lay the framework for systematic testing of relevant hardware and software technologies for the next phase.

SnowBot:

To understand SnowBot, it is best to think of it as a snow blower with a human approximation of a human brain telling it where to go, what to do and when to do it.
SnowBot expands on the traditional role of a snow blower, in that it navigates and guides itself in an alien terrain without the aid of human input and clears the terrain of snow as effectively as a snow blower. With the use of statistical learning algorithms, SnowBot "learns" and "predicts" the behavior of static objects, objects at rest, and dynamic objects, objects that are moving, that maybe present in its immediate vicinity. This in turn, allows it to determine regions of interest, defined as areas where snow has piled beyond a certain threshold height, and then move to these regions, while "learning", "predicting" and actively avoiding objects in its environment.

From the guidance and navigation perspective, the Snowbot is functionally similar to an autonomous car. Therefore, the following question naturally arises**:**
If SnowBot is similar to an autonomous car, why does the project team hold the stance that it can deliver on SnowBot, whereas companies like Tesla, operating at economies-of-scale and with thousands invested in R&D, have yet to develop Level-4 autonomous cars?

The answer to this question lies in the domain in which the SnowBot operates in. Unlike an autonomous car, the range of speeds in which the SnowBot functions in allow it to be more lenient in terms of the requirements in levies on the algorithms that govern its function. To put it simply, the SnowBot, by virtue of travelling at a speed much lower than a car, has more time to react to dynamic objects in its environment; if a deer jumped in front of a SnowBot, it would need a dramatically smaller stopping distance than a car would. Moreover, the penalty of making a "bad" decision is much higher for a car than it is for a SnowBot; a person can be seriously injured by a car going at a speed of 5 miles/hr whereas a SnowBot travelling at the same speed

would be inconsequential. This aspect of the project **reinforces the probability of project success and allows the stakeholders to estimate a lower project risk for this project than with a similar project aimed at car.**

Further bolstering the case for lower relative project risk is the controlled environment in which SnowBot will operate. Unlike the plethora of worse-case scenarios associated with an autonomous car, the worse-case scenario of a SnowBot are mostly technical malfunctions that rarely put human lives at risk. The ability to design SnowBot with a particular environment allows the algorithms to be less involved and yet still be versatile and robust due to the limited variety of environmental contexts in which a snow blower used.
In particular, the SnowBot was developed to navigate itself and clear snow in a residential garage, house or shed. As part of its initialization routine, a programmed set of instructions to be executed every time a SnowBot is booted up, the SnowBot will determine the perimeter of the area and remember it for every use in the future as a target area from which snow needs to be cleared from. Subsequently, the SnowBot will proceed to determine the aforementioned regions of interest and move to one of these areas, while avoiding static and dynamic obstacles such as cars, people, mailboxes, etc. Once the SnowBot has determined that it has cleared all of the snow in the area, it will return to a pre-specified landmark start/end location.

If completed, the SnowBot would find a market consisting of traditionally public entities, like counties and districts, who would employ them to clear snow in public areas, and also private entities, who would employ them to do the same but in space owned privately. In essence, the SnowBot would benefit its potential users in one of several ways:

- Time and money would be saved in human labor.
- Bureaucratic inefficiency would be saved as SnowBot would require much less oversight
- SnowBot can be routine to predict snow-storms and thus, would be better equipped to handle it and take preventive measures to ensure reduced snow accumulation
- SnowBots can be used to initiate a localized response to excessive snow accumulation that would be more effective than the current existing method.

Among the many startling statements made in the booklet titled **Snow Booklet** written by CSU faculty, Nolan J. Doesken and Arthur Judson, the most notable can be found in the following statement:
Despite huge efforts to clear snow and ice, tens-of-thousands of traffic accidents and personal slip-and-fall incidents still occur that claim lives and generate hundreds-of-millions of dollars in medical costs.
Although a decisive statement cannot currently be made as to how impactful the SnowBot would be in combating the issues outlined in the aforementioned statement, one cannot help but be

optimistic about how a decentralized approach like the one advocated by SnowBot could not only furnish a greater economic utility but also serve a greater social need.

Design Description:

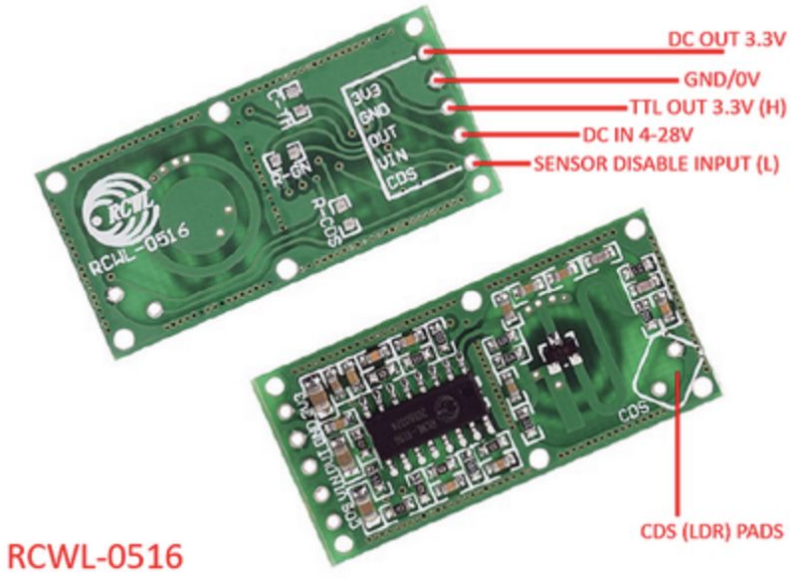
Hardware:

IR

The acquisition of this piece of hardware led to the determination that it was not suitable for this project. This is due to the limit imposed as a result of low visibility from snow. As a result the team has decided to drop the use of the IR sensor from the project and instead use a combination of radar and a camera in combination for the sensor system.

Radar

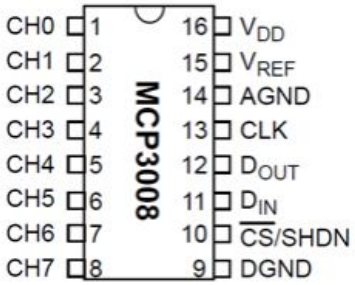
The Snowbot is using a RCWL-0516 doppler radar microwave motion sensor module which can act as an alternative to PIR motion sensor widely used in burglar alarms and security lights. It detects movements within its detection range by using a microwave doppler radar technique to detect moving objects. The sensitivity range of the radar is 7 meters.



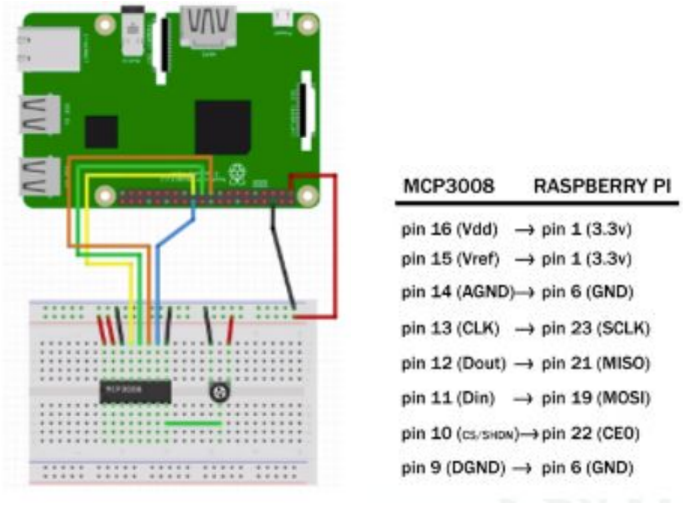
The radar has many applications, some safety measures are required to follow for smooth transition. During construction, avoid any metal part in front of the sensor module. For appropriate results always keep a minimum of 1 cm clear space in the front and rear side of the module.
When the sensor detects the presence or a motion, it outputs a trigger of high voltage 3.3v otherwise it 0v, when printed through terminal or a compiler it outputs digitally with the help of an analog to digital converter.

MCP3008

The MCP3008 is a 10-bit Analog-to-digital converter which converts the Analog input to digital output. Communication with this chip is accomplished by using a serial interface compatible with the SPI protocol, which means to have it working the PI needs to have it's SPI setting enabled. It is programmable to provide four pseudo-differential input pairs or eight single-ended inputs. For the SnowBot since we are just using the chip for a single radar we are using one channel (CH7).



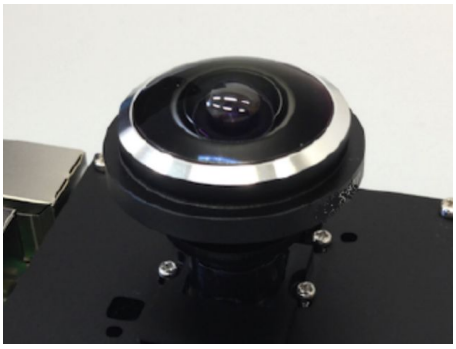
Below is the wire map of how the converter is being used in Snowbot for appropriate results.



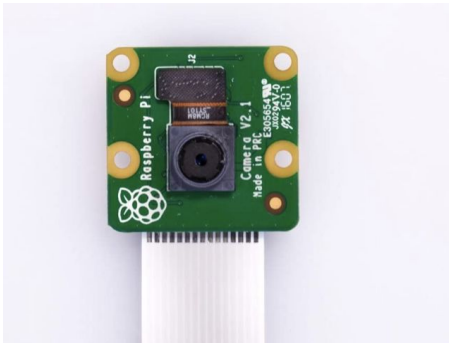
The MCP3008 device operates over a board voltage range of 2.7V - 5.5V. Therefore Pin 2 and pinn 4 on the raspberry pi will be the appropriate input voltage.

PiCam360

SnowBot uses a camera to identify human presence over radar for more in depth information. The PIcam 360 is a 360-degree panoramic camera which provides 235 degree image vertically and 360 degree image horizontally. To plug and use the camera instantly, the camera is using a public library called fswebcam which allows raspberry pi to use the camera through USB. Further implementation of the library import of fswebcam is in the software section of the report.



If needed, SnowBot can also work with normal PIcam, with the same software on a low level Object detection for the next phase.



Software

All software for this phase has been committed to a GitHub repo:
https://github.com/snowbot-dev/snowBot

The Exact implementation can be found at the link above. This section will focus on a high level explanation of how each sensor is read in software with only some implementation details. For a quick guide on how to run the code please refer to the **Testing** section of the report.

Radar

Sensor/radar_sensor in github

The radar sensor is read using python. The python libraries of Adafruit_GPIO and Adafruit_MCP3008 are used in order to read the analog signal from the device. In order to implement this system code was borrowed from the open source repository T04-MCP3008 by makersdigest on github. This code reads the signals from the GPIO pins and stores the output into a variable. As natively Raspberry Pi's GPIO pins are used to collect digital inputs these libraries, along with the circuit mentioned in the hardware section, are used to convert the data into an analog representation.

Code Idea:
The code will collect data over a desired interval and runtime (in seconds). These will be given as inputs and then the data shall be stored into a list. The list is then saved to the data directory as a csv file which contains the interval at which data was collected at index 0 and the collected data from index 1 onward.

Plot

Plot.py in github

After data is collected from the radar and stored in a csv file Plot can be used to graph the data. This is done by using matplotlib, a commonly used python library, to plot a line graph of the analog value vs time.

Code Idea:
By reading the csv file that was created from the radar we have all the information we need to construct the graph. Since our implementation of the radar stores the interval at which data was collected at the first data point we can seperate the first index from the rest of the data (the list of the data will now be referred to as y). Then we can make a list the length of y with each index being an "interval" value greater than the last (this list will now be referred to as x). For example if the interval is 2 and the number of data points is 5 then x will be [2, 4, 6, 8, 10]. We now know all the values for our x and y axis. From here we can use matplotlib to create a plot of the data points.

PiCam/PiCam360

For just a test with a PiCam camera_test.py in github

The Picam as well as PiCam360, although PiCam is a 360 camera, is handled quite simply. To complete this task we use the fswebcam module. Once installed we simply need to run a command to take a picture. This command is fswebcam <filename>. To be implemented with the rest of the system we will have to take a picture at the same time as a value from the radar is saved. Since PiCam works on the same code if PiCam360 were to ever be an issue in the future, the regular PiCam would be a suitable replacement.

Integration

To test the the integration of the sensors run driver.py or driver_360.py in GitHub

Although both files should work for both PiCams however we do have 2 separate files for the regular PiCam and the picam360. However we recommend PiCam360.py for both.

For integration the driver files are a mixture of the radar_sensor, camer_test and Plot python files. These will run for the runtime and collected data at the intervals specified. When collected data it will read the analog value off the RaspberryPi's GPIO pins and then take a picture with the connected camera.

After data collected is finished a plot will be formed of the collected radar data. After completion the data dir inside the project will contain the csv and graph of the radar data as well as the images collected at those same intervals.

Note: as of now only one camera can be used at a time and a camera connected to the motherboard takes priority over one connected through USB

Testing:

What follows below are instructions on how to run the code for the project.

Radar

Testing only the radar can be done with the following steps:

1. Navigate to the Sensors directory within the project
2. Run "python3 radar_sensor.py"
3. You will be asked for what interval you want to collect data over. Given a number value

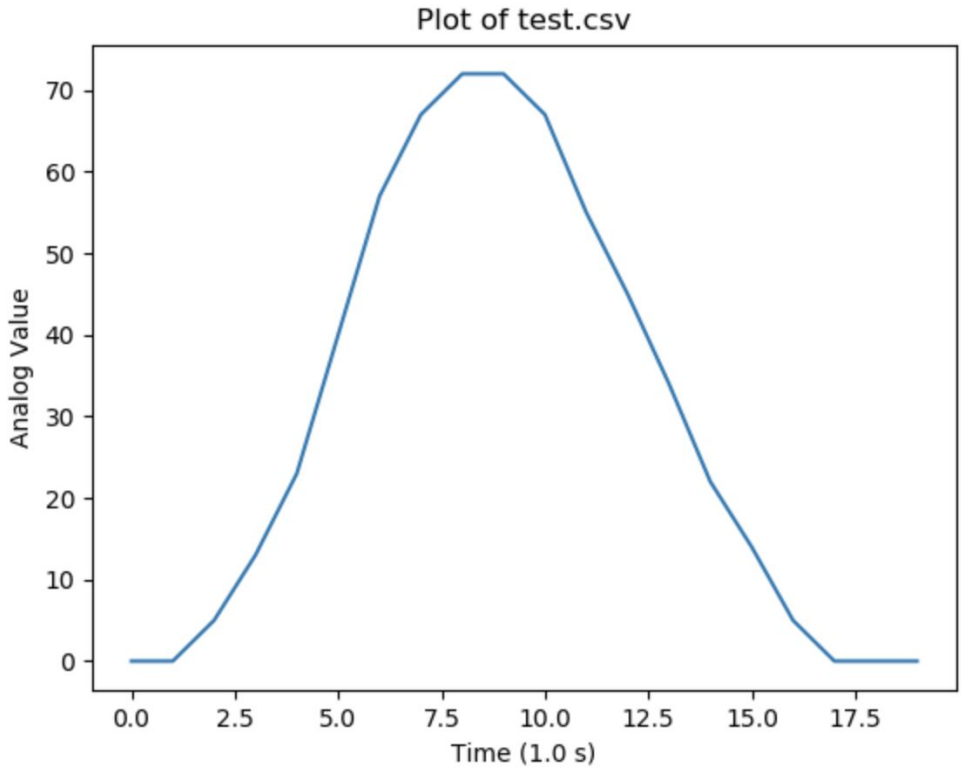
4. You will be asked for the total runtime of your data collection. This number must be bigger than your interval and it is recommended to make it a multiple of your interval
5. The values collected will be printed to the screen, after which you will. Be asked for a name for your file. The filename should end in .csv. For example “test.csv”
6. Your collected data can be found at root/data/<filename>. With <filename> being the name you gave for the file and root being the project directory

Note if you want to collect data indefinitely put -1 for the runtime. This will make the code run forever. If you do this you must use Ctrl-c to stop data collection. As of now if you use this method you will not be able to have your data saved.
Note if you want to collect data from radar and plot you can navigate to the project directory and run “python3 Radar_Plot.py”. Then responded to the questions in the terminal. If you get stuck you can refer to the steps above and for Plot.

Plot

To create a plot of the data collected from your radar follow the following steps:

1. Navigate to the project directory
2. Run “python3 Plot.py”
3. You will be asked what file you’d like to plot (this file should be within the data directory). For example if you want to plot test.csv then put “test.scv” and if you want to plot date/csv type “data/csv”
4. You will be asked if you want the plot to be saved (Saved to a file). Type your response
    a. If yes they will where you want the image stored. All you need is a filename to store in the data directory however if you want it stored in another directory inside of data you must specify (e.g. date/test.png)
5. You will be asked if you want to show the plot (show in a new window). Type your response
    a. If yes to saved and to show, you must close the window before the plot will be saved



Above plot is shown as an example, which was taken from our radar testing phase. It was taken in a time interval of 1 second and a total time taken was in 20 second.

PiCam/PiCam360

Testing only the PiCam/PiCam360 can be done with the following steps:

1. Navigate to the project directory
2. Run “python3 camera_test.py”
3. Image will be save in the project directory with the name “to.jpg”

Both Sensors

Testing both the radar and PiCam of choice

1. Navigate to the project directory
2. Run “python3 driver.py” or “python3 driver_360.py” (driver_360.py recommended)
3. You will be asked for what interval you want to collect data over. Given a number value
4. You will be asked for the total runtime of your data collection. This number must be bigger than your interval and it is recommended to make it a multiple of your interval
5. The values collected will be printed to the screen, after which you will. Be asked for a name for your file. The filename should end in .csv. For example “test.csv”

6. Your collected data can be found at root/data/<filename>. With <filename> being the name you gave for the file and root being the project directory
7. You will be asked what file you'd like to plot. You should put the exact same filename you gave for step 5
8. You will be asked if you want the plot to be saved (Saved to a file). Type your response
    a. If yes they will where you want the image stored. All you need is a filename to store in the data directory however if you want it stored in another directory inside of data you must specify (e.g. date/test.png)
9. You will be asked if you want to show the plot (show in a new window). Type your response
    a. If yes to saved and to show, you must close the window before the plot will be saved

Plans for the Next Phase:

Objectives to be attained at the end of Phase 2 testing are as follows:

- Explore how the data is transformed by the algorithms
- Established concrete details for the system and began the prototyping phase.
- Formulate controlled tests and interpret results to draw a conclusion and improve in future steps from that conclusion.
- Create a testbed for SnowBot.
- Perform a day-in-the-life run using real sensor inputs.
- Establish testing benchmarks following the aforementioned tests.

Conclusion:

The team feels very confident in our progress with the project so far, and we are extremely excited to continue working towards a final product. Snowbot has gone from a simple idea on paper, to a now fully functional system of sensors that have readable output. We feel confident that over the next several phases of design and prototyping we will achieve our goal to create a real and working autonomous system capable of saving people time and money. Snowbot has the potential to further the field of autonomous applications through the implementation of sensors and development of algorithms, and the team is thrilled to be a part of such a new and changing field.