EMBEDDED SYSTEMS

EMBODIED AGENTS, DIGITAL CONTROL IN A PHYSICAL WORLD
WORLD ROBOT OLYMPIAD 2014

Jesper Madsen – sobuno@cs.au.dk – 20101783
Jakob Bjerre Jakobsen – jakjak@cs.au.dk – 20102095
Henrik Knakkegaard Christensen – knakke3@cs.au.dk – 20082178

June 26th 2014

Our code can be found on github
`https://github.com/JakJak7/Lego2014/tree/master/exam/code`

Indhold

0.1 WRO day 1

Date: May 19th 2014
Duration: 9-16
Group members: Henrik, Jakob, Jesper

Goals for today

First thing we will do is work out a threat analysis for the World Robot Olympiad 2014 challenge.

Threat analysis

- Positioning, how will the system know where it is
- Strategy for turning solar panels
- Strategy for replacing solar panels
- The system must be able to pass solar panels without moving them
- How will we pick up a solar panel to move it around
- More importantly, how will we transport a solar panel back between the correctly positioned ones without touching them?
- Keeping track of points on the robot
- Error handling, what will the robot do if it loses track of its position? Important to not keep going and possibly lose points
- Detecting state of a solar panel; the color sensor returns black if nothing is seen, how will we distinguish black from nothing?
- Motor limitation, we can only have three motors
- Handling, can the robot turn around without issue?

In the official challenge, there is a bridge on the track. This is not present on our test track, and so we will disregard it in our project.

Navigation

Goal

Our main focus today is to come up with a strategy for how to navigate the track.

Plan

- Brainstorm ideas for how to navigate the track.
- Discuss suggestions with their pros and cons.
- Decide on a method to use in our final solution.

Possible solutions

- Tacho counter (Cartesian coordinate mapping).
- GPS + compass.
- Line follower.

GPS

We could use an ordinary GPS sensor to find the system's position and a compass sensor to find its orientation. This is a very robust solution, as the system can, at any time it becomes disoriented, just stop and read its exact position, and it is independent of lines to follow, tacho counter etc. It is possible to get an accuracy of about 2 cm using L1 GPS [1].
This accurate GPS module takes 12 and a half minute to find the position, and as we have a strict 2 minute time limit, using this method for navigation is infeasible. Alternatively we could use conventional GPS modules, but these are only accurate down to 3 meters, and as that is about the size of the track we are driving on, it will not do us any good. In addition to this, the compass sensor will be useless if there is any electromagnetic interference nearby. Also, the track is inside a building, which heavily limits the effectiveness of a GPS module.

Tacho counter

We could use the tacho counter to keep track of how far we have moved and use this knowledge to keep track of our position in a Cartesian coordinate system. We can also find our orientation from the tacho counter. The motors are slightly inaccurate however, and after driving around for long enough, our position will start to become inaccurate [2]. And there is no way to correct this inaccuracy, so this solution is not very error resilient.

Line following

The whole track has black lines that we can follow using a light sensor, and we can find our position using certain points and crossings on the track.
Issues we might run into using this method is calibrating the light sensor to properly recognize black, white, green and gray, as we will not know the exact lighting conditions at all times. Another issue is positioning the light sensor, as we usually position right about where the solar panels will be on the track, and we cannot move these or we will have points deducted.

Conclusion

Out of these three proposals, line following is the only one that appears feasible, and so we decide to continue with this approach.

Positioning light sensor

We have a couple of different ways to place the light sensors. - We can place it near the surface of the track at an angle, so the light sensor can be far enough to the side to avoid hitting the solar panels. - We can have the light sensor point directly down, but placed in a height that allows a solar panel to pass underneath.

We will do some measurements to see which solution is better at distinguishing black and white. To do these measurements, we have written a program to save readings for easier comparison.

Results

First result for each color is with the red light on, second is without

Color	3 cm - normalized	3 cm - raw	tilt - normalized	tilt - raw
Green	44	459	50	517
Green	44	458	50	515
Black	47	488	53	544
Black	47	483	53	548
White	53	544	56	583
White	52	541	56	581
Grey	48	495	51	523
Grey	47	486	50	514

Tabel 1: Results for each color in the map (with light sensor)

Conclusion

As seen from the values, both approaches are viable solutions. From the datalogger output, we see that the tilted approach yields more accurate results, but we are worried that the sensor might bump into the panels. The lifted approach is also simpler to construct, and so we decided to go with that.

Color sensor, checking current state of a solar panel

Goal

Get the system to distinguish between a solar panel in the correct position, wrong position, a defect solar panel, or no solar panel at all.

Plan

The way we are going to test if the system can distinguish between the states of the solar panels is by holding the color sensor in the position we expect it to be in in the final design and reading values on different solar panels.
And to clarify, we expect the color sensor to be placed underneath the robot, a bit off to one side. This will allow a solar panel to pass by without bumping into the sensor and thus subtracting points from the final score.

Results

We see that by placing the color sensor as described above and reading values for the four different states, we can easily distinguish between the states. Thus, this is a good way to position the color sensor, and it will allow us to distinguish between states.

State (color)	Red	Green	Blue
Working (blue)	527	536	576
Turned (red)	647	581	583
Broken (black)	522	515	518
Empty (none)	742	741	741

Tabel 2: Results for solar pannels different state

Conclusion

- We decided on a way to navigate the track.
- We found a good way to position the light sensor so that the vehicle can follow a line.
- We found a good way to position the color sensor so that we can distinguish between working, defect, missing and turned solar panels.

References

[1] J. A. Farrell, T. D. Givargis, and M. J. Barth, "Real-Time Differential Carrier Phase GPS-Aided INS," in " IEEE Trans. Contr. Syst. Technol., vol. 8, no. 4 pp. 709–721, July 2000. 1991. CA, Jan. 1998, pp. 345–354.

[2] Lesson 9 lab report

0.2 WRO day 2

Date: May 22th 2014
Duration: 10-16
Group members: Henrik, Jakob, Jesper

Goals for today

Come up with a way to manipulate solar panels.

Plan:

- Make a list of requirements for turning and moving solar panels.
- Brainstorm ideas for how to satisfy these.
- Discuss proposals.
- Decide on design of mechanism.

List of design requirements:

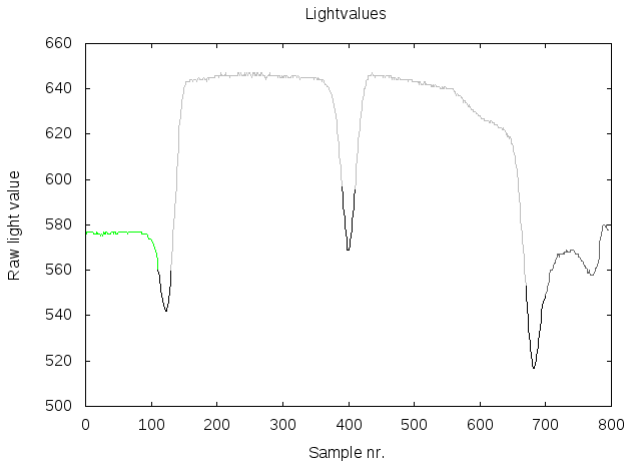
- Must be able to turn a solar panel 180 degrees.
- Must be able to identify working, broken, misplaced and missing solar panels.
- Must be able to hold onto a solar panel for replacement.
- Must be able to carry a solar panel past other solar panels on the track.

Ideas:

- Turn a solar panel by rotating the entire vehicle.
- We have to make sure the solar panel is in the center of the vehicle, or it will move outside of the designated area.
- No third motor is necessary, however if we want to also be able to move the panel around, we will need some mechanism to hold onto it. Even if we add some mechanism for grabbing onto the panel, we will run into trouble later if we need to move it around, as it will be in the way of other panels on the track.
- Have a separate mechanism for rotating a solar panel, running on its own motor.
- Here we will also be able to move a panel around by turning it 90 degrees to lock it under the vehicle. Then we can push the panel around, but again we will run into trouble when we need to get past other panels on the track.
- Lift up the solar panel, turn the entire vehicle, and put the panel back down.

- It will have a mechanism for holding onto a panel and lifting it from the track. This might be difficult to implement using only a single motor, but if done properly it will let us pass other panels without touching them. One way to do it would be to take advantage of the gap in the solar panels and use a thin rod to stick in the gap. The problem with this is that we need incredible precision in order to hit the target, but if we can follow the lines accurately enough, it may be possible.
- Vehicle with a truck bed for storing defect panels; the vehicle will collect all the black panels so it will not have to worry about them later on.
- This will make it easier to bring in the replacement panels.
- It will need a mechanism for lifting up the panels, perhaps we could implement this and the above proposal together, which, while difficult to implement, satisfies all the requirements.
- We will have a mechanism that pushes a solar panel to the side, into a slot where it will be held onto. This way it will not be in the way when passing other solar panels and can be easily transported. To put the solar panel back into position on the track, we will simply put the mechanism in reverse, and so it only requires one motor.

We decided that the best solution will be to lift up the solar panels. It will solve all of the requirements, and if we can implement it properly, we will have a good shot at completing the objective. However, the precision required is worrying, so first off we are going attempt to make a precise line follower.
We need it to able to hit within 2mm of the gap. We wrote a basic line follower to test, but we immediately ran into issues with recognizing different colors.
We tested the light sensors ability to distinguish colors by letting it start



Figur 1: Light reading for a simple run from the base to the garage in a straight line (with the light sensor 3cm above the track).

in the green starting zone, drive out across the white area, crossing a single black line, more white, and finally into the gray storage area. As seen, green is about the same value as black, which might cause trouble later on if we need to determine which area we're in. One solution would be to replace it with a color sensor, but we'll return to this if it becomes necessary.

Conclusion

We decided on lifting solar panels and storing defect ones in addition to the line follower. This will make the foundation for our first prototype, and now we need to start building and testing. However, we immediately ran into trouble when we started testing the precision of the line follower. We will continue the testing next time, and depending on the result, we may change strategies.

0.3 WRO day 3

Date: May 27th 2014
Duration: 12-16
Group members: Henrik, Jakob, Jesper

Goals for today

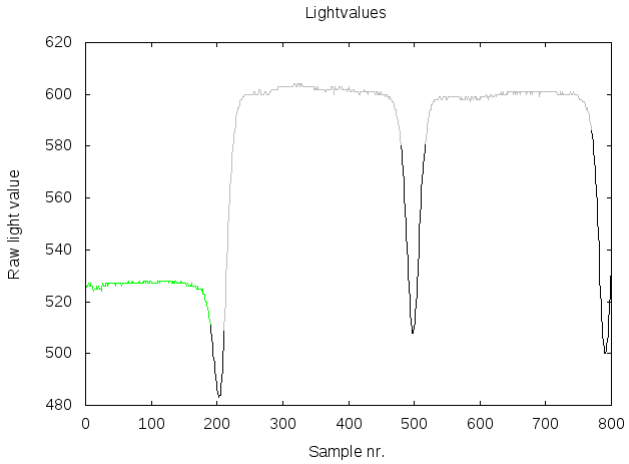
Today we want to have a prototype build ready and get some testing done.

Plan

- Build and realize our design decisions from the last two meetings. We already have a basic line follower bot built, but as seen last time, we need to calibrate it to distinguish colors.
- Test if the designs can actually use the lines to navigate the track, so we can reliably pick up solar panels.
- Make adjustments to accommodate for unforeseen situations.

Results

We have found out that placing the light sensor 3 cm above the track and having it point straight downward did not give the results we got in our initial tests. Perhaps this is because the light source has changed in brightness. In any case, we changed light sensor position to being closer to the track, at an angle. Here are the readings:

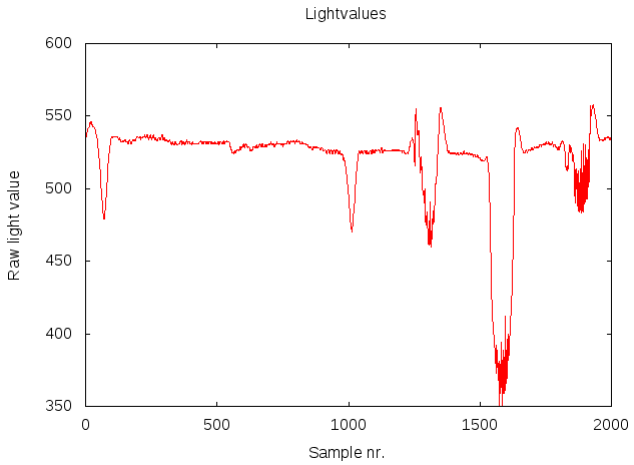


Figur 2: Light reading for a simple run from the base to the garage in a straight line (with the light sensor tilt)

Here it is somewhat easier to tell green and black apart, so we will experiment further with an angled light sensor.

Further testing reveals very inconsistent readings. From using the datalogger with more experiments, we see that following a black line reads values ranging from 425 to 540, with dips when we reach an intersection. At these crossings, more black is seen, and so the light value is lower than normal. Still we are able to see that this is not white, which reads about 600, so we should be able to get the line follower working. Another issue is that currently the light sensor is too close to the line and it bumps into the solar panels sometimes. We have to adjust its angle to move it further away from the center.

We only tested the ability to distinguish colors using the 3 cm above method, we never got around to test line following capabilities. We then tried going back to positioning the light sensor 5 cm above. In this experiment, we had the robot drive straight forward, following a black line. From the readings, it is easy to see when we are on a line, at an intersection, at a solar panel and if it is broken.



Figur 3: Light reading for a simple run, where the robot starts a the cross section inside the station and then follows the line out into space.

The first two little spikes are from intersections, the next wider spike is from a colored solar panel (notice the upward spikes from the gray area before and just after the panel), then there's a huge spike from a black solar panel, and finally another colored one.

Directions: We have assigned cardinal directions to the track in order to more easily describe it. Moving out of the base is north, moving from the green starting zone to the warehouse is west, and the remaining two cardinal directions can be deduced from this. Driving "north": 602 and "south": 607, "east": 560 "west": 620. We get different readings in different directions because of the windows in the building that are facing "west". We also get different

readings in the same direction but at different positions on the track, because the light is not uniformly distributed on the track. This makes it very difficult for us to use line following in practice, and we may have to scrap the idea entirely. The readings may get more precise if we move the light sensor closer to the track, so we return to doing experiments at 3 cm distance. Back at 3.4cm:

Color distinguishing

Clear distinction between green and white, black and white, but the black line measured in the white area reads about the same value as the green area. It will suffice if we are clever about it, but it is not ideal.

Line following:

When following a line, it is very easy to see when we meet an intersection. The light value drops about 50 units. Seeing a working solar panel has about the same effect, but we will still be able to use this to navigate the track. A broken solar panel is very easily recognized, as the light value drops about 200. We are also able to tell T sections from 4-way intersections, as these differ 20 units.

When moving in different directions on white, we read values between 570-630. This is quite a large range, but seeing as black is about 540, it should not be an issue to tell them apart.

After initial testing of the line follower, we found that the single light sensor is not accurate enough, so we will be using two light sensors. We will position the two sensors at a height of 3.4 cm above the surface, 4 cm apart so there is room for a line between them.
Now we have the issue of two light sensors reporting different values on the same surface. There is roughly a 50 unit difference, so we need to make a mapping so that they are equal. If this is anything like the week we used two sound sensors, this will not be an easy task. The mapping could be non-linear and impossible to predict.
We appear to have some success calibrating the two sensors by subtracting their difference from the higher value, and thus equalizing them. This is assuming there is a constant offset, which is unlikely, but we did manage to get the vehicle to follow a line using this approach.

Conclusion

Due to all of there sources of disturbance (lighting in the room, difference between sensors, inaccuracy due to distance from the track), we conclude that it is unfeasible to use the solar panel manipulation approach we decided on last time; lifting solar panels using some sort of lifting mechanism. We simply cannot get the desired accuracy of 2 mm on the line follower.

We have to postpone the solar panel manipulation mechanism until next meeting, so we did not reach the goal for today. We decided to continue with the mechanism that pushes a solar panel aside into a slot for carrying, which we will build next time hopefully.

0.4 WRO day 4

Date: May 28th 2014
Duration: 12-16
Group members: Henrik, Jakob, Jesper

Goal

Due to the troubles we had with distinguishing intersections in order to navigate, we figured it may be worth our while to scrap the line follower approach entirely, and try navigating through the tacho counter. This will be quick to implement, and if it's able to stay on track, we will have saved a lot of time. And so, our goal for today is to explore the possibility of using a tacho counter, whilst trying to improve the line follower.

```
1   public static void forward(float distance){
      int degrees = (int)(360*distance / (wheelSize *
         Math.PI));
3     Motor.B.rotate(degrees, true);
      Motor.C.rotate(degrees);
5   }

7   public static void right(){
      double degrees = 360*((width*2 * Math.PI)/4)
9       /(wheelSize * Math.PI);  //wheel rotations
           required for a 90 degree turn of the vehicle
      degrees = degrees * 1.05f; //taking slight
         inaccuracies into account
11    Motor.B.rotate((int)degrees/2, true);
      Motor.C.rotate((int)-degrees/2);
13    }
```

Using these two methods, and a similar method for left(), and measuring the track in centimeters, we were able to get the vehicle to cover the whole track in a short time. After the first couple of turns it became quite inaccurate though, and we quickly saw the downside of driving using only the tacho counter. In addition to that, the vehicle we have is rather flexible due to a shoddy build, and this makes the wheels slide around, reducing the accuracy of the vehicle.

While the tacho counter method was being developed, a line follower was being written and tested simultaneously. We tried to implement a PID controller from previous weeks to make the vehicle better at line following, but it later occurred to us that since there are no difficult curves on the track, this is unnecessary. Instead we implemented a proportional controller, and the results were promising. The vehicle could follow a straight line without issues.

Conclusion

Implementing a tacho navigator would require us to rebuild the vehicle and spend a lot of time tweaking parameters in order for it to not lose track of its position, and even then it will have no way of correcting its course. Therefore it is not a feasible solution.
We will proceed with the newly implemented P controller, and use that for line following.

0.5 WRO day 5

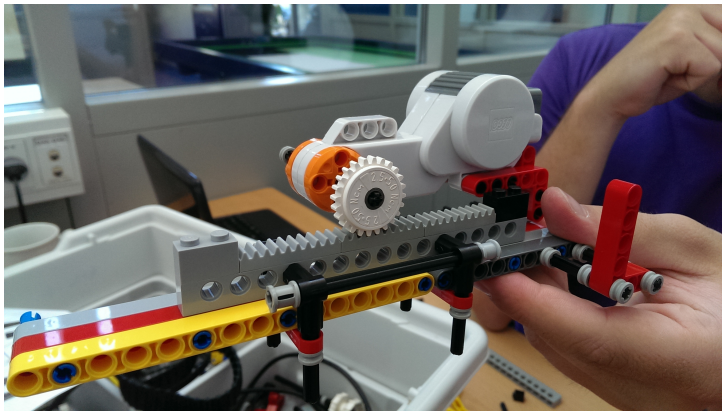
Date: May 30th 2014
Duration: 9-14
Group members: Henrik, Jakob, Jesper

Goals for today

We will continue working on the line follower, while some of the group will start building the mechanism.

We got a P controller for the line follower to work so that it can follow a line. However the vehicle gets confused when it meets a cross section. We'll need a way to handle these sections on the track, so we're gonna implement behavior control.
We'll have a default behavior that simply drives forward using the line follower. When the light sensor picks up lower light values, it'll be overruled by a turning/navigation behavior. When a solar panel is detected, another panel checking/manipulating behavior will take over.

We have the first build of the mechanism ready, it just needs to be added to the vehicle. It's quite large and and will allow for some inaccuracy of the line following



Figur 4: Mechanism

It uses the motor to drive the cog, making the solar panel slot move back and forth, allowing us to lock a solar panel in the side of the vehicle. We need to rebuild the vehicle in order to sport this large mechanism, and once this is done we'll be able to test the solar panel manipulation capabilities.

Currently, our proportional controls have a factor Kp = 1.0, and it's able to follow a line quite well.
`https://www.youtube.com/watch?v=PJHLTqwCBCI`

We have no need for integral or differential control as there are no sharp or difficult turns on the track. We use the tacho counter turns from last meeting to do 90 degree turns, as this is a very reliable way of handling single turns, and after a turn we'll use the line follower to navigate again, thus finding our way back onto the line. This is a very promising way of following lines, and as seen in the video it works very well until we hit the gray areas with solar panels. We've discussed using a color sensor instead of the light sensor, which will help us overcome the obstacle that is color recognition at variable light levels, which caused us trouble previously.

Conclusion

We have an early version of the mechanism built, and a new vehicle is being built around it. Our old vehicle is able to follow a line and turn quite well, which we'll need to migrate into the new model.

0.6 WRO day 6

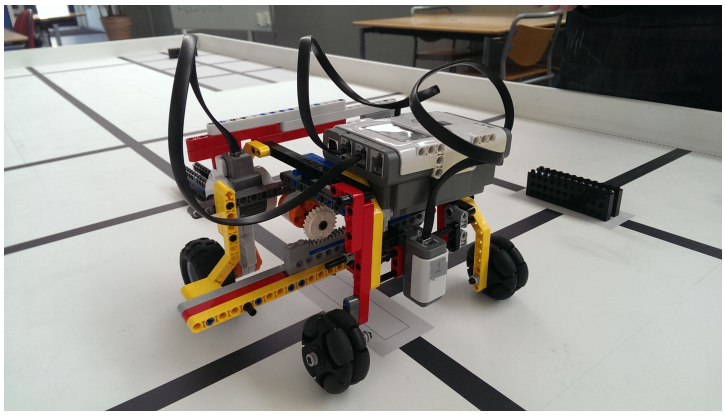
Date: June 3th
Duration: 9-12
Group members: Henrik, Jakob, Jesper

Goals for today

- Finalize new robot prototype build
- Have it follow the track using the line follower + tacho turner
- Get a first run going!

Results

Today went with building the new prototype, which we have ready for action. The mechanism is working fine, although we still need a way to hold onto the the solar panels.
We have the robot following lines, and rotating almost 90 degrees.



Figur 5: Picture of our prototype.

Conclusion

Todays meeting came to an early end as one member had to go, and we decided to postpone the full track run for tomorrow.

0.7 WRO day 7

Date: June 4th 2014
Duration: 8-16
Group members: Henrik, Jesper, Jakob

Goals

- Get the robot to recognize cross sections
- Get the vehicle to rotate solar panels
- Get the vehicle to move around solar panels

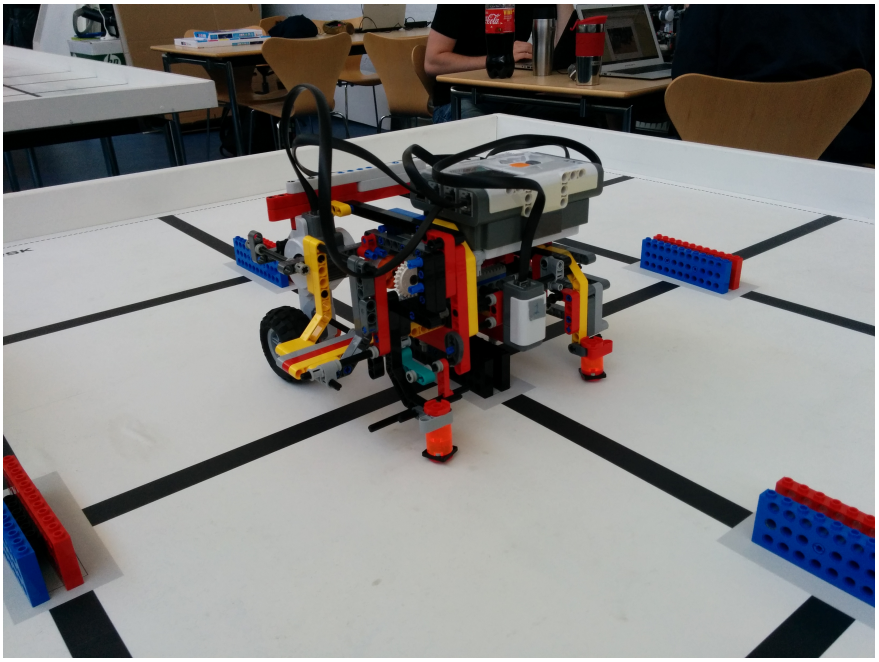
Results

We keep having inconsistent turns, even using the power of mathematics doesn't do the trick. We tried moving the front wheels further apart to see if that would make the turns more accurate, but it didn't help. Instead, we replaced the front wheels with these smooth pads that allow the vehicle to glide around with little friction. We decided to use these because we suspect the omni-directional wheels we used to use generate too much friction, and make our turns inaccurate. The pads were inspired by a kind group of engineers working on the same project.
We have also expanded the grabbing mechanism so that it's easier to catch a solar panel, and now the whole panel manipulation mechanism is working quite well. We installed the color sensor today, for registering what state a solar panel is in. Initially it couldn't register colors, and saw everything as black, because of the distance to the panels. We'll write our own calibration for it.

We've been told that you can increase the accuracy of the light sensor by placing a brick with holes in it in front of it. From an initial test, we see that truly this is the case. However, positioning it correctly using only lego has proven to be incredibly difficult, due to the alignment of the holes. Therefore we have not included this in our vehicle.

Suggestions for easing the problem

- Make the track longer, giving our vehicle more room to turn. Our vehicle is a bit large, and when we rotate in order to turn a solar panel, we sometimes hit the adjacent panels.
- We'd like more time for the challenge. Two minutes is simply way too little to both check the solar panels and rotate them, and to replace the defunct ones.
- Have fixed lighting conditions, it's extremely difficult to to calibrate the light sensor when the lighting keeps changing, differs from direction to direction, etc.



Figur 6: Our prototype without front wheels.

Conclusion

We still need to take care of the cross sections, but little by little we have every part of a full run we need. The color sensor code has been written, but needs testing.

0.8 WRO day 8

Date: June 5th 2014
Duration: 9-15
Group members: Henrik, Jakob, Jesper

Goals for today

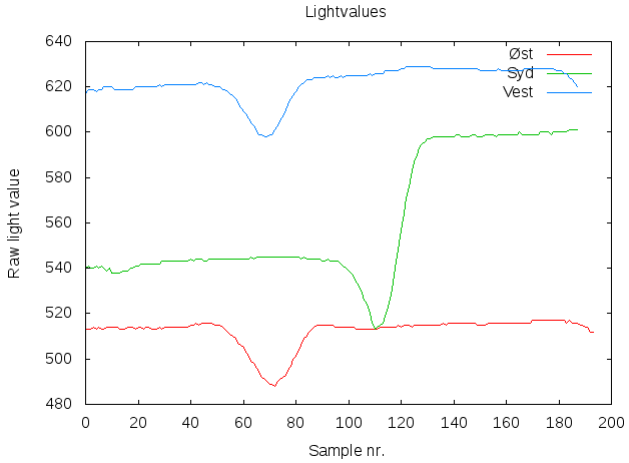
- Recognize the intersections
- Get the vehicle to ride the whole track

Plan

Do measurements of light values when reaching intersections and T intersections from all four directions (there's no T section in that one direction).

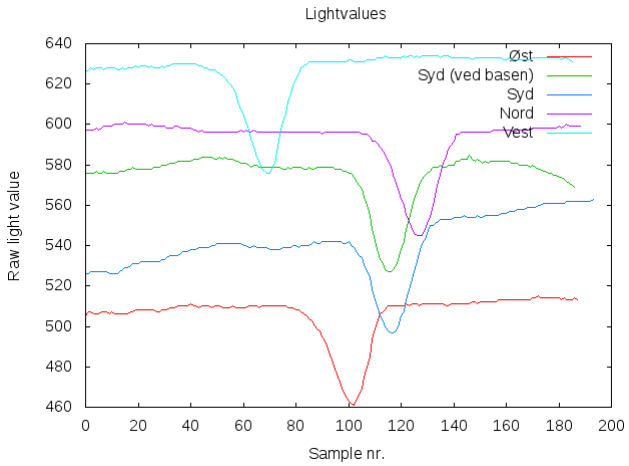
Results

The way we measured the light values when crossing T intersections is by enabling the datalogger, and letting it follow a line to a T intercourse. As



Figur 7: Light values for three different directions when the robot follows a line and see a T-section.

seen in the graph, the offsets are quite different, but the change in values is very consistent, about 25 units. The syd graph increases after the intersection, because it entered a white area.
We tested the same way for 4-way intersections, Again we see a quite consistent change of values, about 50 units. These tests show that it is feasible to tell apart 3-way intersections and 4-way intersections, although the values are relative to the direction the vehicle is pointing.



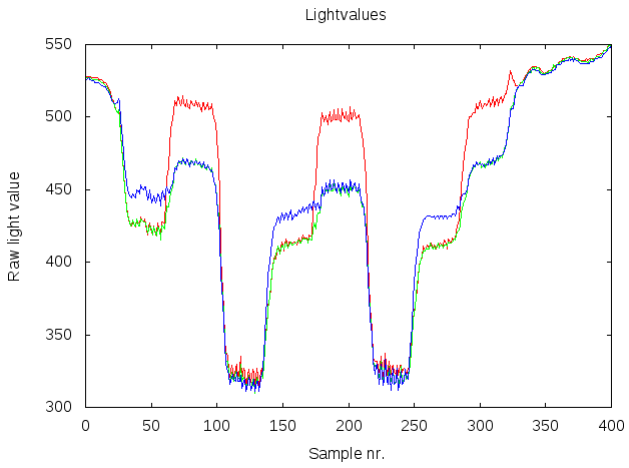
Figur 8: Light values for three four directions when the robot follows a line and see a cross-section and one for a different direction.

We realized we haven't tried driving backwards with the vehicle, so we tested that.
`http://youtu.be/DYHlgq6TYgY`
As seen in the video, it is not possible to drive backwards using the P controller, as the light sensor is still on the line even though the robot is getting way off track.

Turns out we need another light sensor in the back if we wanna be able to drive backwards, and so we shall install one!
After several hours of naming variables for light values, we have the second light sensor installed. The program starts out by calibrating the light sensors, by driving out into the white area, reading values, turning around, reading values from another direction, and finally the vehicle starts following the line towards the solar panels.
We're now working on getting the vehicle to follow the track through each line and turn, and once we have that running, we can use the solar panel manipulation mechanism we developed last time to finally earn some points.
We're able to make the first turn, but we have issues getting the second turn to work. The light sensor cannot reliably detect the T cross, and so it keeps going. The lighting in the room appears to be the cause of this trouble.
After hours of fiddling around with parameters, we found a bug in the code, and now the vehicle also makes the second turn. We're now well on our way to the solar panels, so we can finally get some manipulation done! `http://youtu.be/hDwU0r5dlgE`

Turns out the solar panel turning method no longer works accurately, as our vehicle has been built wider, and the panel is no longer in the center of it. We had to make adjustments in order to center it again, and now the vehicle

is able to catch a solar panel and rotate it again.
The color detection method we wrote yesterday was based on static lighting conditions, and this is not a feasible solution in practice. Therefore, we're rewriting it to dynamically calculate the color thresholds on runtime.



Figur 9: Color mesurements when the robot drives over 8 solarpannels in a straight line (none,blue,red,black,blue,red,black,blue,red,none)

From the RGB value graphs, we see that in order to determine if there's a panel or not, we only need to look at the blue, or even green, color value. More on this next time!

Conclusion

We didn't get the vehicle to finish the track today, as we ran into trouble getting the turns right. Hopefully next time we'll get more turns working, and finish the track. The color sensor still needs some calibration.

0.9 WRO day 9

Date: June 6th 2014
Duration: 11-13
Group members: Henrik

Goals for today

- Discover solar panels and their states (color)
- Make the robot align with the line nicely so it may hit the solar panels correctly
- Create a testing program using hard coded values and experiment with this

Discover solar panels

In the previous session, we made quite a few measurements for the color sensor when a red, blue or black solar panel was present (or none at all). We made these measurements multiple times under different lighting conditions. Based on this, we have found a way to disover the different states the solar panels may be in.

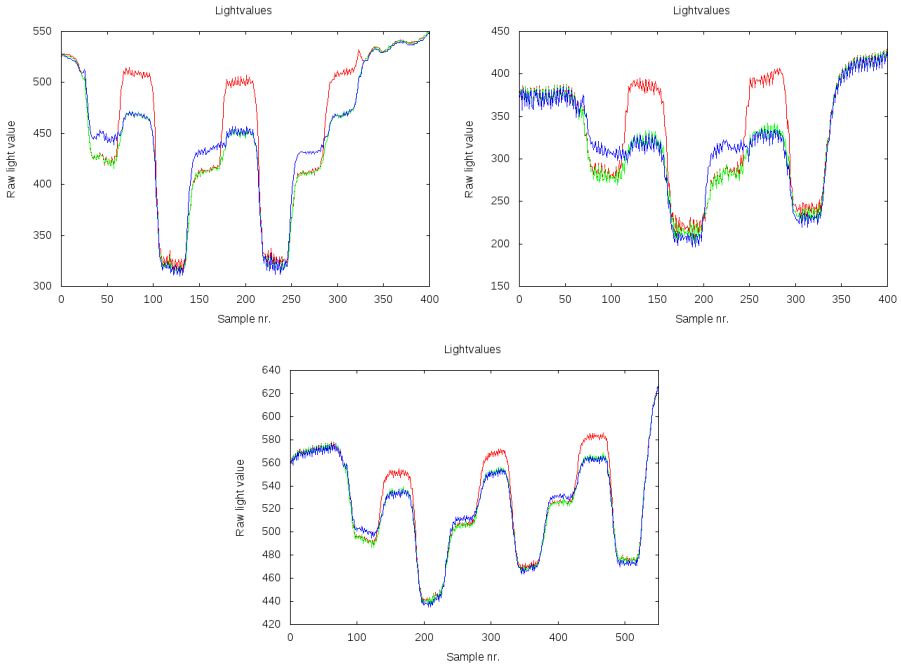
First idea for discovering solar panels

On every graph, it is clear that all the RGB-values are high when no solar panel is present, but when a panel is present, the values change. When a solar panel is present, we see on our measurements that the green value is always among the lowest, so we are going to make a function that takes the value when there is no solar panel (ValueNone) and subtracts the value for green (ValueGreen) and checks whether the difference is large enough to determine whether a solar panel is present.
htp

Detection color	Mesured values for the three runs			equal to this offset		
Blue (max-green)	525-425	375-275	570-490	= 100	100	80
Red (max-green)	525-475	375-325	570-530	= 50	50	40
Black (max-green)	525-325	375-225	570-450	= 200	150	120

Tabel 3: Mesurement for the three graphs.

As can be seen in the calculations above, red is very difficult to discover as the difference needs to be greater than 39 (approximately), and this was also reflected in the test run we made using these calculations.



Figur 10: Three color mesurements when the robot drives over 8 solarpannels in a straight line (none,blue,red,black,blue,red,black,blue,red,none)

Second idea for discovering solar panels

The values for when there is no solar panel present (ValueNone) are all very large and there is very little difference between them. We are going try to incorporate this, so that when the RGB-values become very small, but the difference between them is still very small, we can conclude it is a black solar panel. In the same way, when the difference between them is large, it's either a red or a blue solar panel and if the value between blue and green is small then it's a red solar panel, and if the value between red and green is small then it's a blue solar panel.

This approach has worked on 2-3 tests for each color, so we are satisfied with this idea at the moment.

Hitting the solar panels correctly

We have a problem with the robot's direction being a bit off when the last turn has been made (when it should be on the line with three solar panels in front of it. The back wheel is not aligned with the track, so even though we are using a P-controller to control the direction of the front of the robot, it will always be a bit off, and as a consequence, we will "always" hit the solar panels a bit off. As such, we need to align the robot before we can attempt to

pick up a solar panel.

First idea

Drive straight forwards and backwards until the backside light sensor hits exactly on the darkest part of the line we want to follow, then turn the front part of the robot onto the line also. In connection with this, we need to consider whether the front light sensor needs to be centered with regards to the robot or whether it should continue to be aligned a bit to the left side of the robot.

Conclusion

We made a function for discovering solar panels that we are reasonably satisfied with using the RGB-values from the color sensor. We started on an idea for hitting the solar panels correctly, but did not finish. We will continue with this in the next session.

0.10 WRO day 10

Date: June 7th 2014
Duration: 12-16
Group members: Henrik

Goals for today

- Make the robot align nicely so that it may hit the solar panels correctly
- Create a testing program using hard coded values and experiment with this

Hitting the solar panels correctly

First idea from yesterday

Driving back and forth to align the robot with the line we want to follow. The first attempt at this idea from yesterday seemed to fail; the width of the line was too big, so it was a bit random where the robot ended up.

Second idea

After having turned, drive backwards so that the robot has a longer linie to follow before the first solar panel. Use a P-controller to follow the line, this should make the robot align nicely. We have attempted doing this, and also driving backwards and forwards multiple times, but it still seems that the robot doesn't align as precisely as we want it to.

Conclusion

We continue to have problems aligning in a sufficiently accurate way with the line, so we will start out the next session by continuing this work.

Results

Results for today can be seen at the link below, the video shows one of the best runs. `https://www.youtube.com/watch?v=E85kgq_EDWI`

0.11 WRO day 11

Date: June 8th 2014
Duration: 17:00-19:20
Group members: Henrik

Goals for today

- Make the robot align with the line nicely so it may hit the solar panels correctly

Hitting the solar panels correctly

By lowering the speed of the robot and using the tacho counter, it was possible to make the robot align, and together with the P-controller, the result came out quite good. We also moved the the color sensor a bit further forward, so that the robot would not turn when it registered the grey area under the solar panel. Our current problem is that when the robot approaches the solar panel to pick it up, it will often accidentally push the solar panel a bit forward, which means the robot will not get a proper hold of the solar panel. Part of the goal for next time will thus be to add LEGO to the robot that makes it possible for us to make the robot get a hold of the solar panel correctly more consistently.

Results

The result for today can be seen in the video below, which shows a pretty standard run. `https://www.youtube.com/watch?v=ZhuCP3WbW8g`

0.12 WRO day 12

Date: June 9th 2014
Duration: 9-20
Group members: Henrik, Jakob, Jesper

Goals

- Get the robot to properly hold on to the solar panels
- Get it to replace the broken ones

Results

Everything failed in our initial run because the sunlight was brighter than usual, and so the robot was unable to find any black lines. We changed the light sensor values to recalibrate it.
We had some inconsistent results turning the first solar panel, as the vehicle would often bump into it and move it out of place. To counter this, we lowered the speed of the vehicle as it approaches a solar panel, and the results got much more consistent.
So, now we can turn a solar panel, and we can pick up a broken panel and carry it back to the start. Now we have a problem; we have to exchange the broken panel for a new one, but we have no obvious way of putting down the broken panel while picking up the functional one.
Immediate solution: place the broken panel somewhere else, and go pick up the new one.

We did come up with a clever solution where we move the broken panel into the storage area in front of the fresh, working one, then use our mechanism to move the broken one aside and grab the new one.
`https://www.youtube.com/watch?v=C0-sibkTmmU`
As seen in the video, the vehicle is able to replace the solar panel and leave the broken one in storage, which according to the rules is the proper way to do it.

Conclusion

We got the vehicle calibrated for the lighting conditions, and got it to replace broken solar panels. Next thing we should look at is getting it to replace several panels in one run, as it currently is only capable of taking the first fresh one.

0.13 WRO day 13

Date: June 16th 2014
Duration: 14-19
Group members: Henrik

Recap & Goals

It has been a week since we last worked on the project due to exams. So far we have made the robot find its way to grid #1 and turn all solar panels on this grid. We have also made the robot find its way to grid #1, spot #2, carry a broken solar panel back to the warehouse from this position and return with a working solar panel. (Grid #1 is the east-most line of solar panels according to our earlier cardinal directions definition)

Today the plan is to see if things still work and then continue refactoring the code, such that the aforementioned functions (Turning and replacing solar cells) continue to work while changing the code from being hard coded to being more dynamic, so that the robot will do the right things no matter how the track has been set up with regards to solar panels.

Do things still work?

We have run our old program, and aside from a few bugs due to the lighting, it worked out. We may need to find a better way to measure the light, but that might mean a lot of analysis on different data, which we might not be able to do in time unless we have an additional month.

Refactoring

The code has been split into more functinos and classes, so that we can more easily call the different commands (Turn robot, move robot, turn solar panel, etc.). We started out by building a program with this new data structure to see if things were still working; the program drove out to grid #1, spot #1 and turned the first solar panel. It worked just as well as it did with the old code.

We then decided to try out a theory which we have been considering for the past week. The theory was to use a light sensor on the backside of the robot to find the spot that the robot's rotation would revolve around. As the light sensor is the positioned on the middle of the axle between the two motors that rotate the robot, the robot will, when the backside light sensor is above an intersection, after rotating be positioned so it is correctly aligned immediately. As alignment has been a big problem earlier, this would be a very nice improvement. We switched out the P-controller with a new function for terminating the movement of the robot; this new function terminates the movement when the backside sensor hits an intersection, at which point we

start rotating. There were a few problems in the initial runs as, after rotating, the backside sensor would still be positioned in the middle of a line, causing the robot movement to terminate again. After a lot of tests and attempts at getting the backside sensor to not make the robot terminate again, we dropped the idea again as we could not get it to function correctly.

We then wanted to get the robot to run faster by doing faster turns and driving faster on the long straight lines. This gave us some problems as our turn method is based on a tacho counter; as we increased the speed from 40 to 70 (Out of a scale of 0 to 100), the robot turned about 30 degrees too much. However, at the straight lines, it worked fine, so we are still able to lower the total run time of the robot by increasing the speed at these areas and then lowering it again for the turns and solar panel rotations.

We should be able to implement a new method for turning the solar panel which can turn it faster, but it will only be in connecting with refactoring the code once the first program for completing the track has been finished.

Conclusion

Today we have refactored the code, so that it will be easier to continue working with, tested (and rejected) a new way of turning the robot (Using a backside sensor), tested code for approaching a solar panel and rotating it as well as made attempts at improving the speed of the robot, which worked in some places.

No major changes in the way the robot looks or works, so no pictures or videos for today.

0.14 WRO day 14

Date: June 17th 2014
Duration: 9:30-22:15
Group members: Henrik

Goals

- Work out a new way of calibrating the sensors
- Work on the code and make it more dynamic
- Write code to make the robot do correct actions for an entire row of solar panels

Calibration

Yesterday we had some problems with the calibration; we have only been calibrating in two directions so far (East and west with regards to our previously defined cardinal directions), so when the robot had to follow a line very accurately in order to hit a solar panel correctly, the calibrated offset value was obtained from an angle that wasn't entirely correct, which means the robot drove in a wrong distance from the line.

Today we have changed the robot to calibrate in all 4 cardinal directions in the hope that this will improve results. When we tested it, it turned out it did indeed improve results, but later on in the day we discovered that even just a small cloud blocking the natural lighting affected our robot drastically. We have not been able to find a solution to this problem unless we find a room that only has artifical lighting.

The code

The code is nearing completiong, although there is still missing an overall structure that implements an overall strategi for the behaviour of the robot. We have started on this and decided that our first strategy shall work as follows:
1. Drive to a row of solar panels.
2. Pass over the row and note whether the solar panels are active or inactive; if they are broken, they will be replaced immediately. This means the robot will pick it up immediately, turn around and drive back to the warehouse, fetch a new solar panel, drive back to the row, find the correct spot, place the solar cell and mark it as active in memory and then continue passing over the row.
3. When the state of all three solar panels have been noted, the strategy continues as below: - If solar panel 3 is inactive, it will be rotated, such that the robot is facing south, which makes it easy for the robot to drive to spot #1 or #2 and pick up a solar panel there. If solar panel 1 is inactive, the robot will drive to this one and rotate is, so that the robot is once again facing

north, ready to rotate solar panel 2 if needed. - If solar panel 3 is not inactive, the robot will drive backwards to solar panel 2. If this one is inactive, it will be rotated and the robot will be facing south, ready to turn solar panel 1 if needed. If this is not needed, the robot drives backwards out of the row using tacho count.

The general idea is to try and minimize the number of times the robot has to drive backwards after rotating a solar panel, which is what is hardest for the robot at the moment.

4. When a row is completed (All solar panels are active), the robot drives to the next row and the process is repeated.

We have also made the code that makes it possible for the robot to replace one broken solar panel and return to the right spot afterwards; there are still a few bugs here, but it almost works.

0.15 WRO day 15

Date: June 18th 2014
Duration: 9:30-13:15
Group members: Henrik

Goals

- Calibration
- Make everything work together, i.e. make the strategy for driving around the track, the strategy for making the robot turn solar panels and the strategy for exchanging broken solar panels work together.

Calibration

As the robot is very sensitive to the lighting conditions of the room, we have to change the calibration method again as we cannot otherwise guarantee that the robot can finish a complete tour if the lighting conditions change (The sun appears, people are casting shadows over the table, a cloud blocks out the sun, etc.). The first plan is to mount a light sensor on the top of the robot pointing upwards, which will only be used to measure the ambient light, both while calibrating but also dynamically adjusting the values we use for the other sensor on the run.

We have noticed when the sun is brighter, there are also darker shadows, so the robot is more affected in some directions than others by this. In order to solve this problem, we either have to shield the robot from the sun completely or make a function that correlates the darkness of the shadows in the room with the amount of light in the room.

Results

After having implemented this and made a few test runs where the light has suddenly fluctuated, it seems it is a reasonable way to fix the problem; there are still a few big deviations from the line, but this seems to be because the sensors measure the light differently (Either because of differences internally in the sensor or because of the location of the sensors).

Strategy

As we explained yester, the strategy for driving is first to drive to a row, fix the row as explained yesterday and then drive to the next row, etc.

As we started implementing this and testing it, we discovered that the function to replace broken solar cells still have some issues, so we are focusing on the inactive solar cells only for now. There were quite a few issues with the robot when it had to turn the solar panels. We tweaked the parameters

back and forth, but nothing helped. Finally, the battery ran dry, and when we replaced it with a new one, the robot functioned without major issues when driving through the grid with all solar panels already active.

0.16 WRO day 15 - part 2

Date: June 18th 2014
Duration: 13-02
Group members: Henrik, Jesper, Jakob

Today

More test runs, it does alright although it often fails to recognize the black solar panels.
We tried cleaning the track in hopes of improving line following, as the track is incredibly dirty at this point. We now have the vehicle display the RGB

Figur 11: Picture of the track, on the right side it has been cleaned, and og the left it has not.

and light values, so that we can read them and hard code them. This way, the vehicle won't have to shuffle around at the beginning in order to get readings, and we'll save some time on the full run.

We saw another group made a paper armor for the vehicle, shielding from light. This is to avoid any changes in the ambient light, and we decided to try this approach on our own.
This didn't help, and actually made our vehicle run worse, so we got rid of it.

Next up we tried moving the entire track to another location with more stable lighting conditions. Here we had the option to turn the lights on or off. Turning the lights off didn't help, because the color sensor registered the floodlight from our light sensor as the color red, and turning off the floodlight isn't an option in the dark.
With the lights on we achieved much better results.

Now we can get the vehicle to turn all panels, and replace the broken ones. As the vehicle was running low on battery, we switched to a fresh battery, and now it would no longer turn nor back up correctly. Battery charge is something we have to keep in mind, now that we've fixed the lighting conditions.
Another thing we've yet to implement, is have the vehicle pick up replacement panels from the three slots in storage. So far we've been taking them all from the first slot, and inserted a new one when necessary. This is illegal in the final competition, so we'll have to make our vehicle obey the rules.

Conclusion

We have moved the whole track to a room with fixed lighting conditions, and thus (hopefully) eliminated the calibration issues we have been troubled with for a while. Currently our vehicle runs great here. Next up we'll get our vehicle to be able to pick up fresh solar panels from storage, that is if we can make it in time, as tomorrow is the day of the presentation.

0.17 WRO day 16

Date: June 19th 2014
Duration: 09-15
Group members: Henrik, Jesper, Jakob

Today

We started out by once again calibrating the robot as the lighting conditions had changed since we left the previous day. We also attempted to expand the method for replacing broken solar panels, so that we could take fresh solar panels from the other two positions in the warehouse. We did not get far in doing this, however, as a lot of people wanted to use the track as it was the day of the presentations. Through the day we continued to calibrate the robot to the ever-changing light conditions before the final presentation. We also sped up the robot when passing by solar panels to improve a bit more on our total run time.

0.18 WRO Conclusion

Conclusion

In the end, we think our project was quite successful. We initially feared that we would not be able to get anywhere near the 2 minute time limit that the official WRO competition has, but we actually have a run with a legal set up of solar panels that only takes 2 minutes and 19 seconds to complete, so we actually made it quite close.

During the project, we have had a lot of issues with the lighting conditions in the rooms we were running our robot in. This is in part due to our choice to try and limit the number of sensor we used; our final robot ended up using only 2 sensors - 1 color sensor for detecting the color of the solar panels and 1 light sensor to follow the lines on the track. It was also partly caused by the large distance our light sensor had to the ground, but achieving this large distance and still being able to follow the lines is also something we are quite proud of.

Future work

The robot is able to complete the track for full points if there is only one broken solar panel (Aside from the time limit). If there are multiple broken solar panels, we are unable to handle more than the first one of them. If we had more time, this is one of the first issues we would be looking at.

We also got some other ideas late in the project for optimizing the runs for the robot. One of these is based on the fact that an official WRO setup will have 4-6 solar panels that are incorrectly configured. This is decided before the run is started. As such, we could implement a counter that increments every time we have reconfigured a solar panel and once this counter matches the number of solar panels that needs to be configured, we could simply return to home. In an ideal case, this would allow us to save precious seconds by not having to check the third row at all.

Another thing we did not have the time to implement was error handling. If the robot malfunctions for any reason, it will continue to execute the entire program instead of stopping. However, as the rules state that the team is simply allowed to call “Stop” to stop the attempt and begin scoring, handling errors by terminating the program is not necessarily. Some form of error correction code could have been nice to have the time to implement however. This could for example involve the 2 sensor ports that we still have open.