TEST PLAN

TEAM: Kurt Weisman, Doga Tuncay
TEST ID: Lab1-1.1 .
DATE: September 17th .

PURPOSE : ***Test*** *square_raw.py and square_sensor.py* ***Tune*** *steering_sensitivity*

CONDITIONS : The square course that we used to test our cart has a straight length of 10 meters with a corner radius of 3 meters. There was an upwards slope from west to east. There was no weather related difficulties during the course ride.

TEAM MEMBERS PRESENT : Kurt Weisman, Doga Tuncay

PROCEDURE : In the first lab we tested the square_raw.py and square_sensor.py by tuning steering_sensitivity and turn_p_term.

The square course that we used to test our cart had a straight length of 10 meters with a corner radius of 3 meters. There was an upwards slope from west to east. There were no weather related difficulties during the test. The square course values were assigned in the .py files under the navigation parameters that are, 'sq_side': 10.0, 'sq_corner': 3.0. The target_heading helped us to determine turning direction with some calculations. It calculated the distance the cart travelled and compared it with the length of the straight line (10 meters), when it travelled the defined distance it turned 90 degrees from the corners.

heading_error = float(compass['heading']) - float(nav['target_heading'])

In this equation we compared target_heading with the cart's current heading to find which direction the cart should go.

Our smallest turn had a 3 meter radius. No matter what the steering correction was, we could not turn sharper than 3 meters radius. The sharpest turn was 3 meters according to some calculations of the Cart's length etc. If we tried a smaller radius than 3 meters, it would harm the Cart, so we stuck with the 3 meter radius.

return steering_correction, heading_error

We returned two values, that is what any embedded systems should return.

We had a control system called 'control' that controls our steering options that are turn and straight. Actual heading shows us that we close enough to target_heading by calculating the error_heading. If this error was smaller than the slope, then it could decide whether to travel 10 meters or not. Then it made its turn.

The navigation parameters are defined below. Steering_sensitivity was our value that is needed to be changed in the square_raw.py code and turn_p_term in the square_sensor.py code.

```
nav = {
      'mode': 'auto',                    # autonomous mode
      'enable': True,                    # Driver enabled
      'sq_side': 10.0,                   # meters
      'sq_corner': 3.0,                  # radius in meters
      'speed': 50.0,                     # fixed for this assignment
      'steering_slop': 5.0,              # degrees
      'steering_sensitivity': 0.0035, # TUNED THIS VALUE #
      'control': 'turn',                 # {'turn', 'straight'}
      'target_heading': 0.0,             # degrees from magnetic
north
      'mark': 0.0,                       # starting position in
meters
      }
```

```
driver_s = {
    'clock': 0,
    'enable': True,
    'direction': 0,
    'side_length': 10, # meters
    'corner_radius': 3, # meters
    'turn_state': 'turn',
    'turn_slop': 5, # degrees
    'turn_P_term': 0.03, # <<<!!! TUNE ME, please.
}

driver_s_doc = {
    'clock': "The clock value on which this data was written.",
    'enable': "True/False - stops reads on driver device",
    'direction': "The target heading of the driver",
    'side_length': "How long a side of the square should be (meters)",
    'corner_radius': "How large of a radius the corners should be (meters)",
    'turn_state': "What state of the turn we are in (turn/straight)",
    'turn_slop': "How close to straight we must be to be going straight",
    'turn_P_term': "The P term of the PID controller for steering",
}
```

After we tuned these values we achieved better and smoother squares. We applied different sensitivities that are shown in the next part.

We observed that, the course was inclined which affected Cart's speed that of which influenced steering. Because of the incline from the west to east we have shifted squares. In other words, inclination affects speed and speed affects the turning point. As you go faster, the vehicle travels further as the steering moves to a new radius.

The current steering sensitivity (0.03) without any changes produced this square :

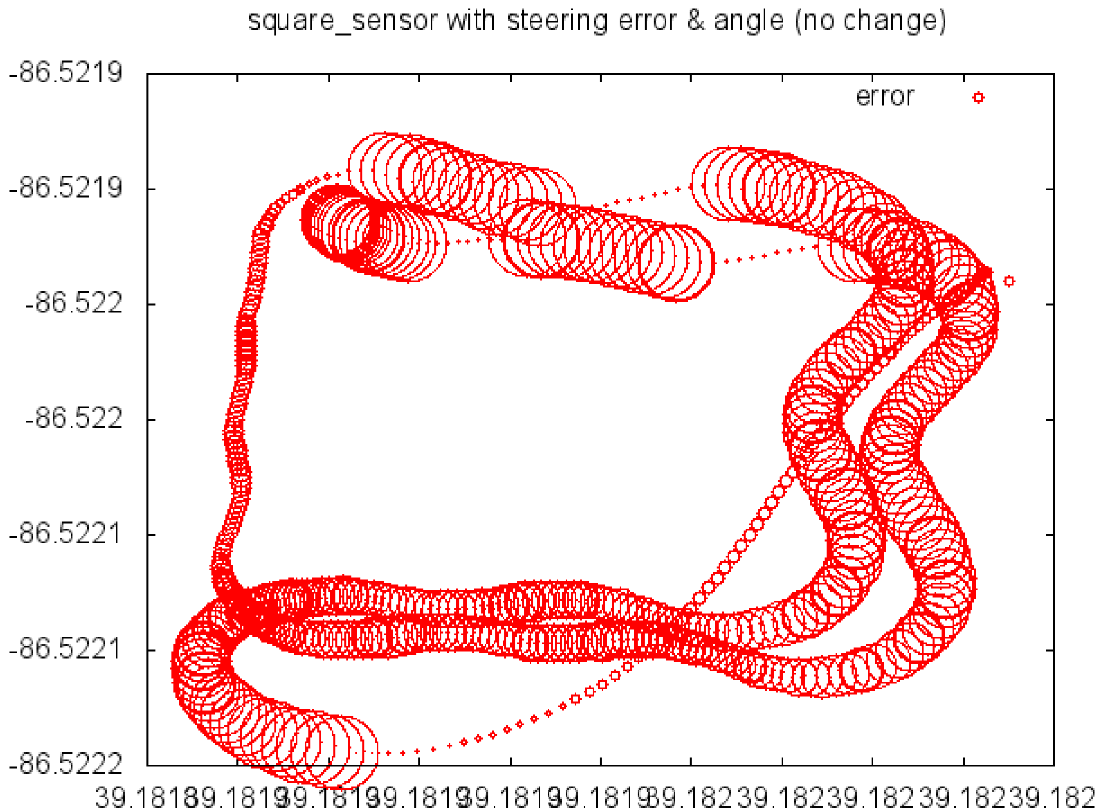

Figure 1.1 No change in the sensitivity

The square is shown above currently is not a smooth square it has more steering error with zigzags.

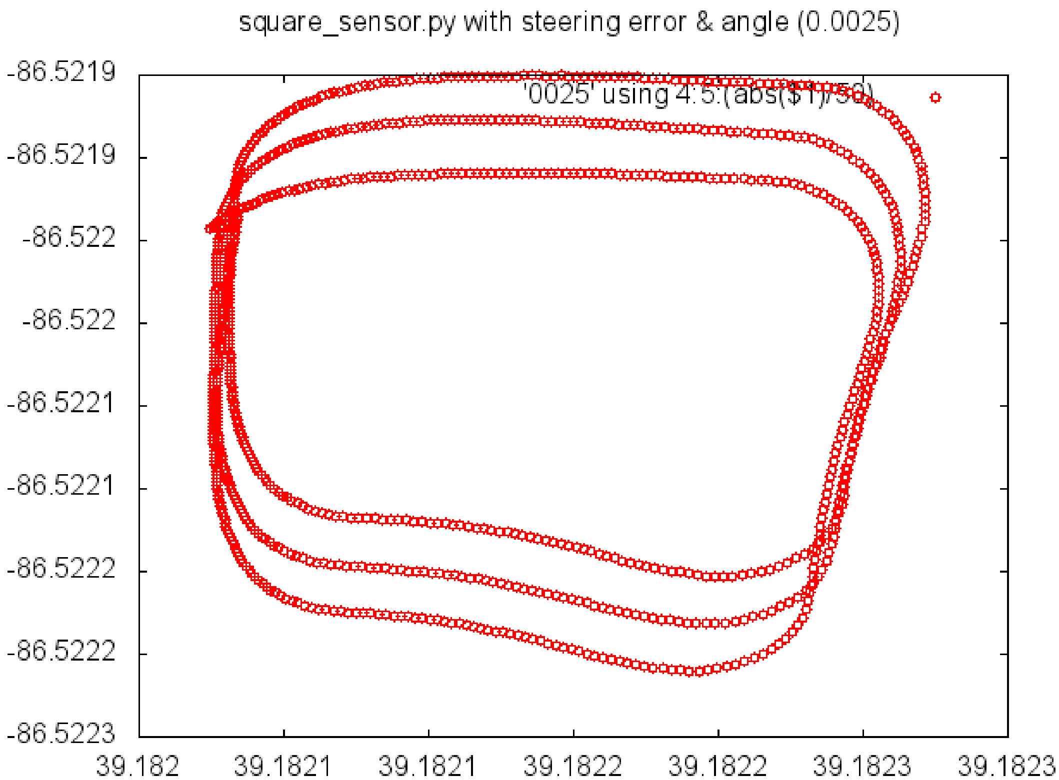

Figure 1.2 sensitivity = 0.0025

In here we had a better square, but we can still observe the shifting which is not something we can prevent.

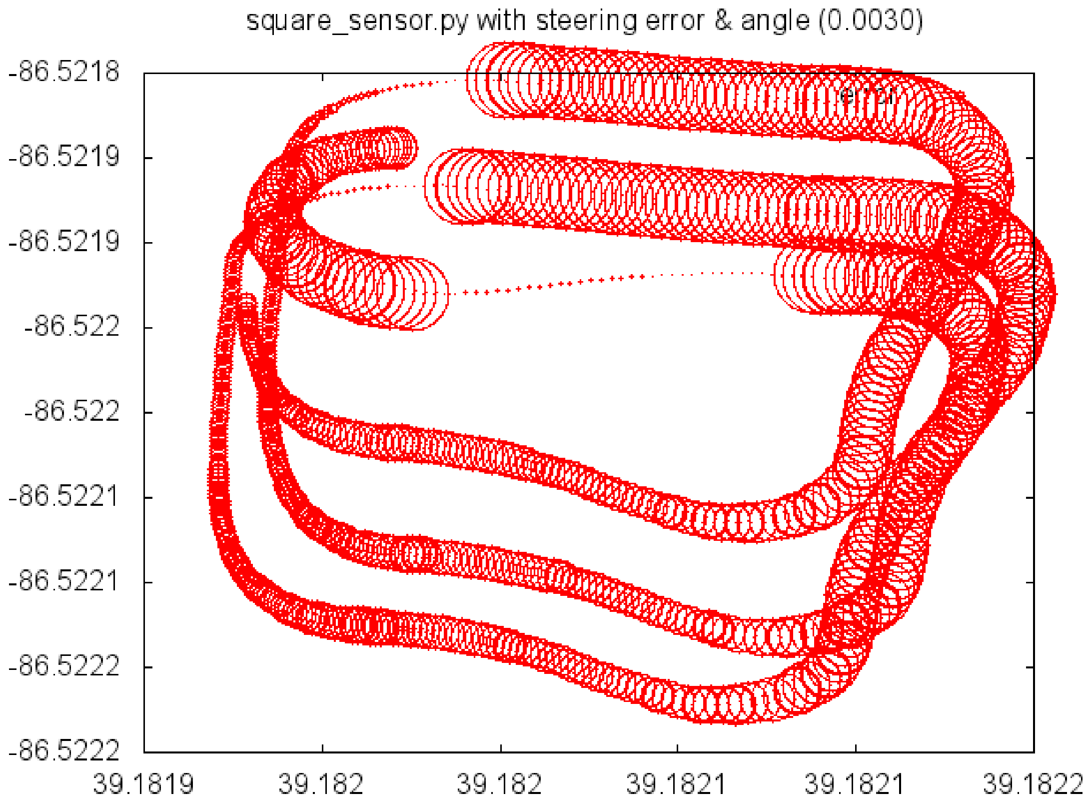

Figure 1.3 sensitivity = 0.003

Finally, the better result is shown in the following graphic which had the better steering and smoother square. As can be seen in all the graphics, we didn't have the same speed too. That is also because of the inclined course that we had. In our first lab, we observed that the best steering parameter is 0.0035.

!!!!

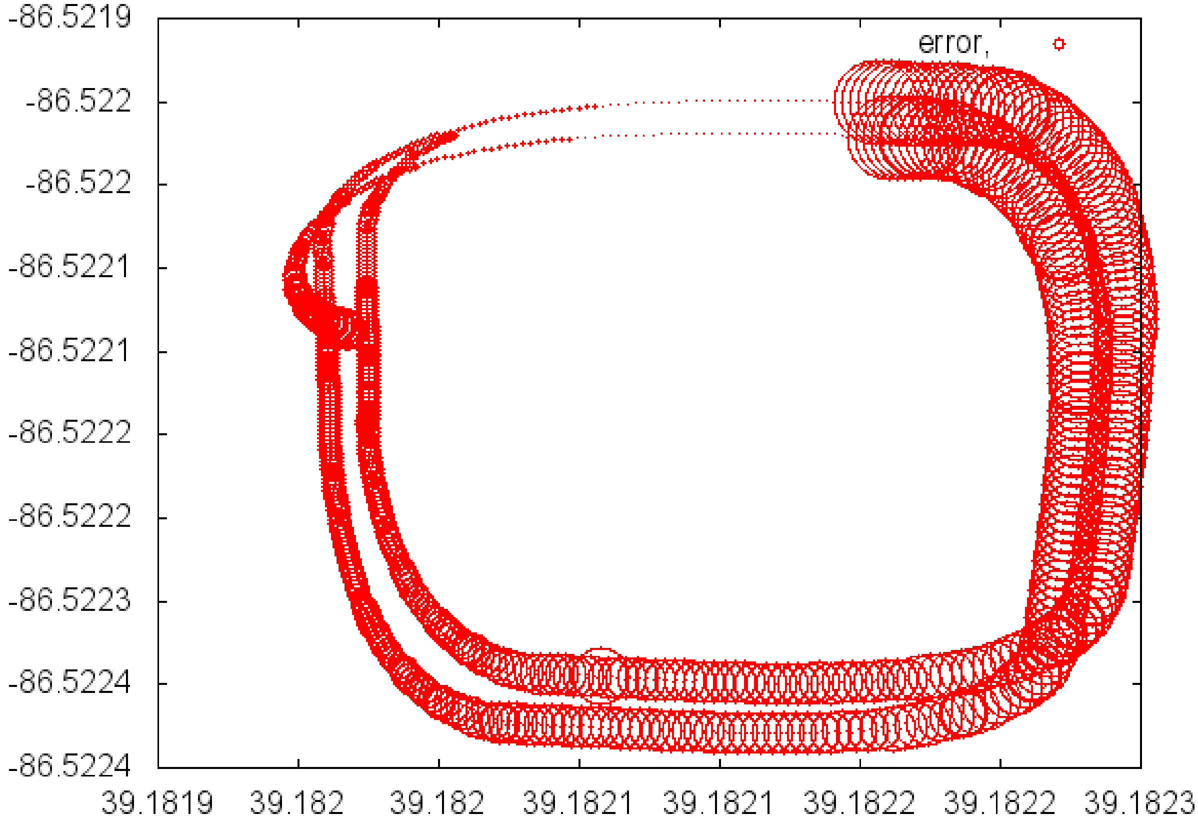

Figure 1.4 sensitivity = 0.0035

We ran square_sensor.py in the simulator by tuning the term_p_term, here are some screenshots:

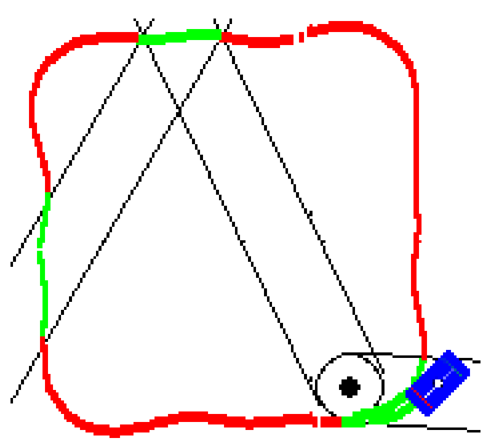

Figure 1.5 term_p_term=0.035

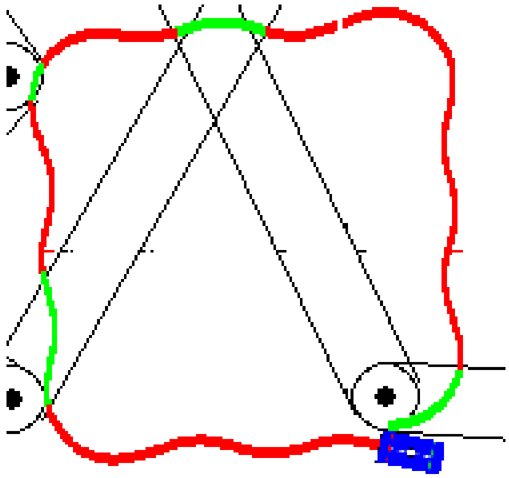

Figure 1.6 term_p_term=0.055

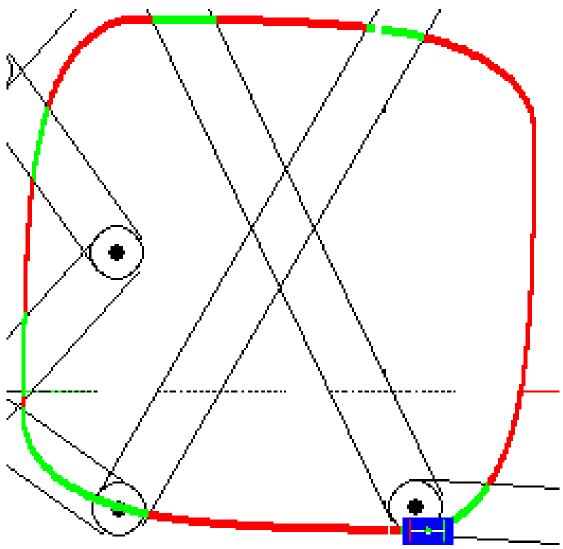

Figure 1.7 term_p_term = 0.0035

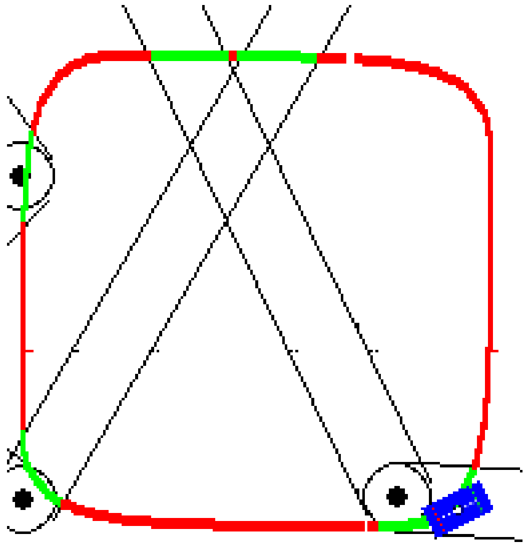

Figure 1.8 term_p_term = 0.0065

As you can see from the screenshots, lower term_p_term we got better squares.