Checkpoint 4 Report

[ICN5406] Mobile Robot 2018

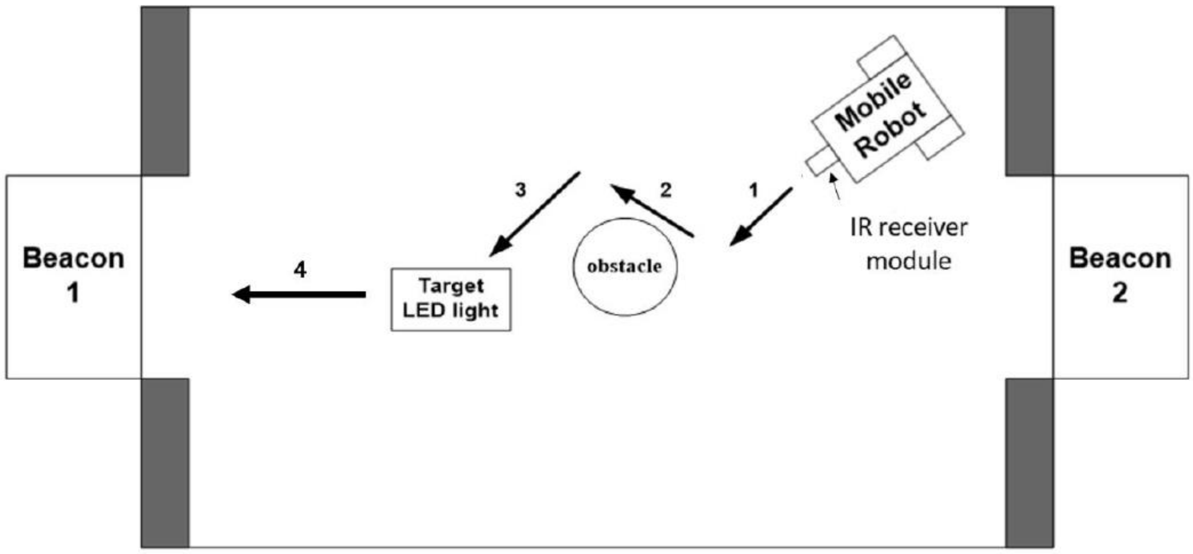
Student ID: 0745011 **Name: Petr Gondek** **Date: 18/12/2018**

1. Purpose:

The purpose of this checkpoint has two goals. First, making sure that our robot can detect a beacon signal and move towards it. Second, combine all the function together for robot hockey contest.

For this assignment, two infrared diodes will be set up at opposite ends of an arena. Each diode will be emitting light modulated at 38 KHz, but their pulse width are different when received by IR receiver module.

We will need to demonstrate our robot's capabilities under relaxed conditions with no other robots in the arena. The arena will be the actual contest arena. As it can be seen on the next picture.

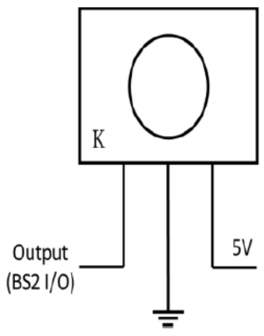


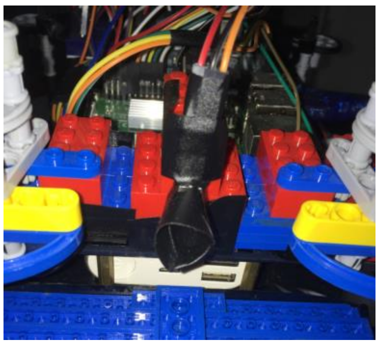
This checkpoint should verify our robot if it is ready to compete with other robot on final robot hockey competition. This checkpoint has to be done in to two minutes.

2. Description of Design:

2.1. Hardware

We got new sensor which can receive the IR signal. We have connected it to the RPI trough blue hub in height of 10.5 centimeters. We made a cover around it, so it can receive only a direct signal. The sensor's output is connected to RPI's GPIO pin.

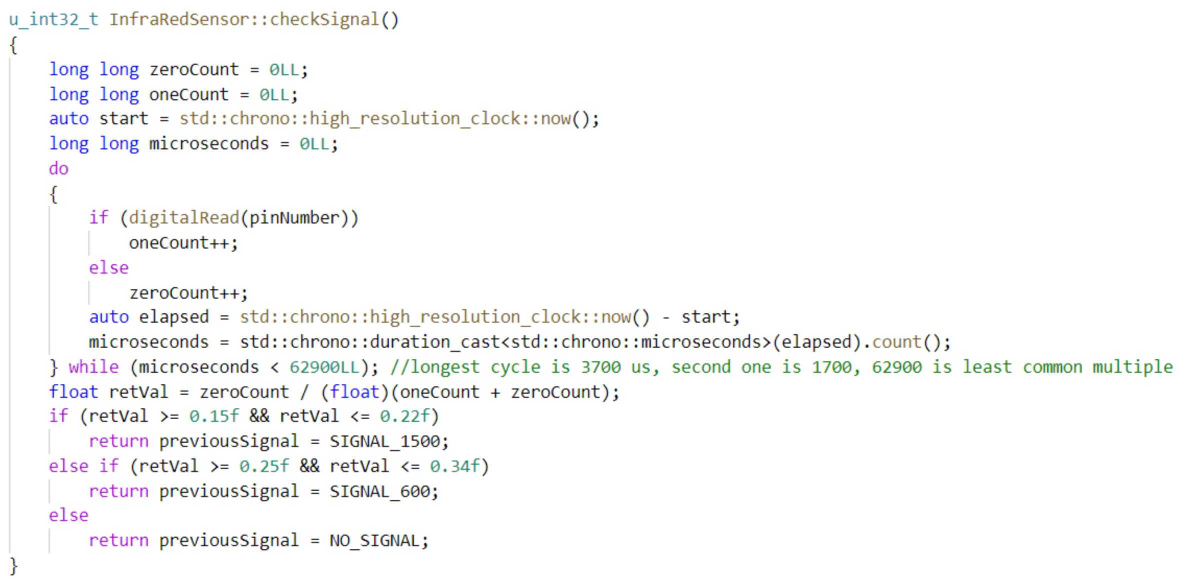




2.2. Software

2.2.1. IR Signal Processing

The IR beacons repeats their signals in 3700 us and 1700 us. Least common multiple is 62 900. That is the time for how long we read the signal and after that we compute the rate. Details can be seen in attached code below. We simply compute rate of received ones and received zeros.

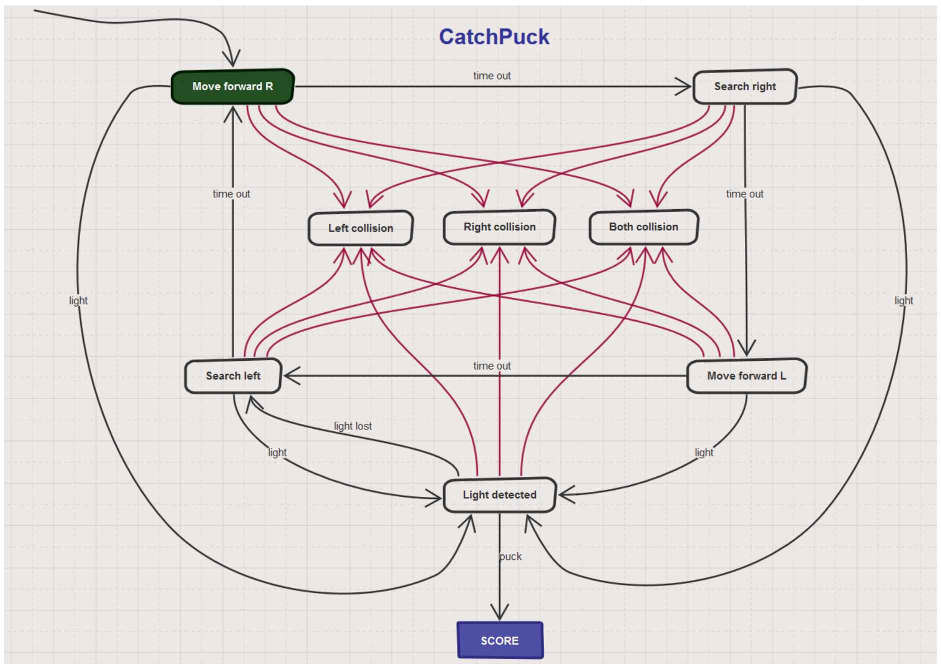


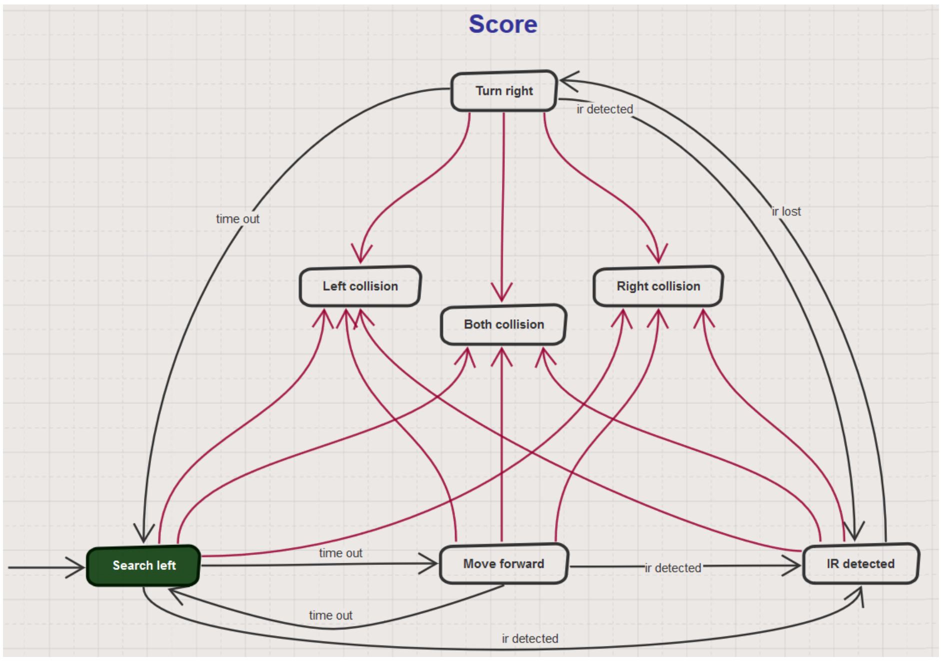
```
u_int32_t InfraRedSensor::checkSignal()
{
    long long zeroCount = 0LL;
    long long oneCount = 0LL;
    auto start = std::chrono::high_resolution_clock::now();
    long long microseconds = 0LL;
    do
    {
        if (digitalRead(pinNumber))
            oneCount++;
        else
            zeroCount++;
        auto elapsed = std::chrono::high_resolution_clock::now() - start;
        microseconds = std::chrono::duration_cast<std::chrono::microseconds>(elapsed).count();
    } while (microseconds < 62900LL); //longest cycle is 3700 us, second one is 1700, 62900 is least common multiple
    float retVal = zeroCount / (float)(oneCount + zeroCount);
    if (retVal >= 0.15f && retVal <= 0.22f)
        return previousSignal = SIGNAL_1500;
    else if (retVal >= 0.25f && retVal <= 0.34f)
        return previousSignal = SIGNAL_600;
    else
        return previousSignal = NO_SIGNAL;
}
```

2.2.2. Finding the IR beacon

Our code is separated to programs and each program contains states. States are exact moves like a MoveForward, MoveBackward, AvoidObsticle and so which defines the program behavior. We have two programs (exactly three, but the third will be explained later). One program finds puck and on event PuckAcquired we switch the programs to second one the ScoreGoal.

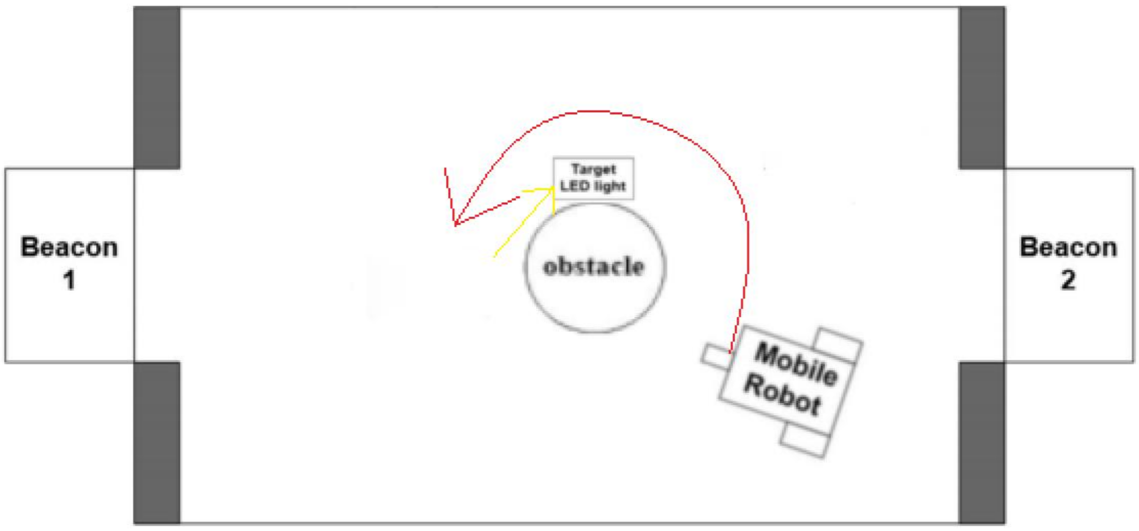
Score goal works randomly. We just go forward little bit and them we look around. You can see details of both programs in next diagrams. Score program is not finite. After solving collisions, we do the SearchLeft again.





3. Result

We successfully found both IR beacons and shot the goal. We had problem with one attempt because the light from puck went from the side trough cover to the sensor. Robot thought that it is directing to puck and went forward but studently pushed puck to the obstacle. The puck was in dead position and we were unable to get it away. For the next attempt we made better light cover for the sensor, so this couldn't happen again. As shown on the next figue.

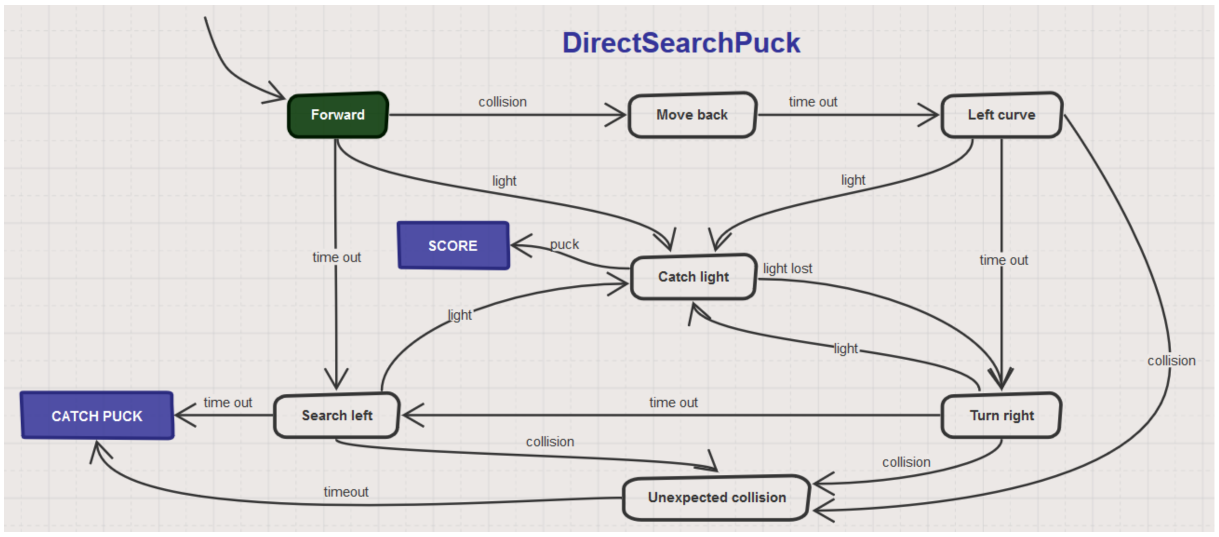


4. Discussion

We discovered that we had wrongly recognized wheel in Arduino program. When we wanted the robot to move left it went to right. This was easily fixed, we just changed the defines in program to right values.

We had troubles to see puck during preparations for demo. We still are not sure what to do with that. We made a better cover for light sensor and we increased the sensor's sensitivity to max possible value that we still do not see the daylight. We don't know if it is enough.

Because we need to be fast as possible, we created new program based on arena knowledge. We suggest that the puck is somewhere on middle line and we start on line in front of our gate. We just move forward to the middle line then we look little bit right and then left. If we did not find puck, we change program to "find puck randomly". If we hit the obstacle, we go around it from the right and we again stop in middle. If the assumptions not work, we change to random program earlier. You can see details on next figure.


DirectSearchPuck
Forward
collision
Move back
time out
Left curve
light
light
SCORE
puck
time out
Catch light
light lost
time out
light
light
collision
CATCH PUCK
time out
Search left
time out
Turn right
collision
collision
timeout
Unexpected collision