CPTR-480
Project Description and Information
rev 1

Introduction

Before being bought by Amazon there was a company that designed and sold Kiva robots that move shelf units stocked with merchandise around inside distribution warehouses owned by Amazon and others (videos of these robots are fun to watch). The Kiva robots travel along what is called a Manhattan style grid, i.e. a grid of “streets” that are parallel in the X direction and parallel in the Y direction and intersect at right angles. A goal of this project is to create simple vehicles that will travel along a Manhattan style grid with capability of traveling from a starting location in the grid to a destination location given starting and ending coordinates with travel between being autonomous.

Vehicle

The vehicle has two wheels each powered by a servo motor that is controlled with pulse width modulated signals that have a 20ms period (50 Hz) and nominal 1.5ms pulse length. Increasing the pulse length from nominal causes motor rotation in one direction and decreasing pulse length from nominal causes motor rotation in the opposite direction. Motor rotation is stopped when pulse length is nominal.

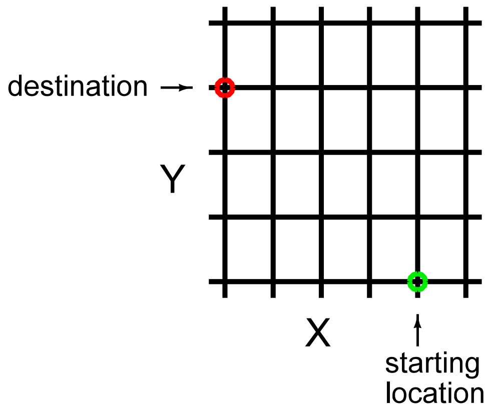
The NXP Freedom board with a rev3 WWU interface board attached will be mounted on the top of the vehicle. An array of 9 optical light emitter/detector sensors will be mounted under the vehicle spaced about 1/8" above table surface. Mounted on the vehicle front will be a US100 ultrasonic distance sensor. An ST Microelectronics VL6180X proximity detector will be also be mounted on the front of the vehicle. The ST Microelectronics LIS2MDL magnetometer will be mounted low on the vehicle front.

Route finding

As mentioned above, a goal is to establish a destination some distance from the starting location of the vehicle, enter the destination information into the vehicle software, and then have the vehicle navigate autonomously to the destination. Because development time is limited, you can choose to enter destination parameters by compiling them into firmware rather than establishing a user interface to enter them. If this hardcoding method is chosen they must be specified using #defines at the top of the main.c file.

An example grid and vehicle locations:

The destination location would be
X = - 4, Y = +3 relative to the start



The coordinate sign will indicate the relative direction and the numeric value will indicate the number of cross-roads away the destination is. An assumption here is that travel will only be in the X or Y direction and that 90 degree turns will be used rather than taking a straight line between start and destination. Suggested parameter names and their defines for this example are:

```
#Xdirection -4
#Ydirection +3
```

You are allowed to define locations in some other way but travel is expected to be along a Manhattan type grid.

Operation Requirements

a) First priority and functional requirement is line following (test on straight and curved lines)
b) Second priority is cross-road detection and 90 degree turn at a cross-road
c) Asynchronous serial data output using UART2 is to be functional and allow output to a terminal window for debugging etc.
d) The US-100 ultrasound sensor is to be active and if an object is reported to be within two grid distances away the vehicle is to change its motion. 1) minimum function: vehicle stops and red LED flashes. 2) optional function: vehicle changes path but continues toward destination.
e) The magnitometer is to be active and if a significant change in magnetic field is detected the vehicle is to change its motion: 1) minimum function: vehicle stops and the red LED flashes. 2) optional function: vehicle changes path but continues toward destination.
f) A stretch goal is to have the VL6180X proximity sensor functional and allow user selection of either the US-100 sensor or the VL6180X sensor to change vehicle motion as described above.

Code requirements

Source code must comply with the requirements defined in the handout posted on the class web page. Comment headers on source files must contain the required information. Each function must be delineated as defined and must show authorship.

Each vehicle has a number. The number of the vehicle your software is set up for and tested on must be included in the main.c file comment header.

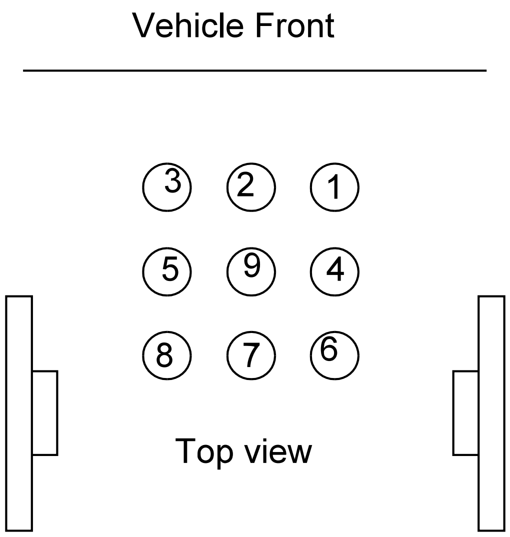
Notes

1) Untethered vehicle operation, i.e. battery operated, is a goal but not a requirement. The instructor will assist in setting up battery operation

Port Definitions

Analog light sensors - see below	
Motor control, PWM	PortC pins 8, 9 (left motor pin 8, right motor pin 9) TPM0 channels 4 and 5
Asynchronous serial I/O	PortD pin 3 UART2
Magnetometer SPI	PortD pins 4,5,6,7 SPI1
US-100 distance sensor	PortA pins 12,13,17 (pin 17 used as trigger, pins 12,13 the echo) TPM1 channels 0 and 1
VL6180X proximity sensor	PortC pins 10,11 I2C1

Light sensor layout - top view



Shown above is the ordering of sensors as if we had x-ray vision and could see down through layers of the vehicle to the light sensors. Left and right are outlines of the wheels. Each sensor (a photo transistor) has a light emitting diode (LED) associated with it, so we don't have to supply an external light source. The output is an analog voltage from each sensor. Here is the signal order:

sensor#	port#	A/D channel
1	PTE20	SE0
2	PTE21	SE4a
3	PTE22	SE3
4	PTE23	SE7a
5	PTB0	SE8
6	PTB1	SE9
7	PTB2	SE12
8	PTB3	SE13
9	PTE29	SE4b

Below is shown the path of the photo sensors when a 90 degree turn is done while keeping one wheel stationary. Note that the front-most row of sensors is located over a cross-road when the turn begins. When the turn ends the vehicles is centered on the cross-road. Sensor locations are shown as little crosses as pointed to by the arrow. The arcs are the path a sensor takes as the turn occurs.