Autonomous parallel parking

ABSTRACT

Self-driving vehicles are now a reality within our society. This report describes a method for an autonomous search for a suitable parking lot and the parallel parking of a two motor model vehicle with the help of an ATMEGA16 microcontroller and a SHARP distance sensor. For this purpose a perforated breadboard with the needed circuitry was mounted on the vehicle. An accurate controlling algorithm to handle each motor independently was needed in this case for a precise parallel parking, so a PWM signal for each motor, with the help of interruptions and the internal timer of the microcontroller, and the use of an H-bridge as well, was generated. This approach was found to be very accurate, and with extra circuitry for noise reduction purpose added, the two-motor model vehicle along with the distance sensor were ready for the search and autonomous parking algorithm. After lots of trial and error algorithms were tested, a careful and efficient autonomous search and parallel parking was programed.

1. Introduction

As technology grows in an exponentially way day after day, the urge to make things autonomous grew. From an automatic light system, to smart houses providing everything you need, the autonomous life is among us: auto-pilot in airplanes, sailboats with auto-tillers, semi-autonomous UAV and even robot vacuum cleaners. Yet self-driving cars have a more complex problem. They have to be able to self-drive themselves in city streets, take over traffic signs, street lights, traffic itself and pedestrians as well.

History goes back to 1920, where Houdina Radio Control showed the world the first radio-controller driverless car "Linrrican Wonder" in the middle of the traffic in New York streets, which was operated by another car through radio signals. Ever since, the race for a perfect self-driving car begun, and this meant the need of an automatic parallel parking system as well. It was not until 2003 when Toyota launched to the market the first car with an automatic parallel parking feature. By 2012 big car manufacturers, including Ford, Lexus, BWM, Audi and Mercedes added an automatic parking system to some top selling models cars.

This report is about a two motor model car with an autonomous parking system programmed in an ATMEGA16 microcontroller, with the help of an H-bridge to control both motors and a distance sensor mounted in a servo motor. The SHARP distance sensor was mounted on a Parallax toddler servo mini F/BB servo motor with an effective range of 180°. Due to that it is possible for the robot to do measurements in front, one side and the back. The servo motor is controlled by a PWM signal from the ATMEGA16.

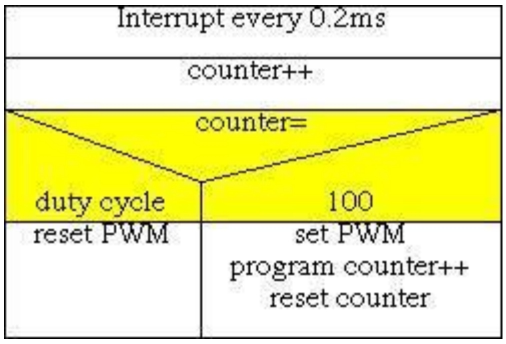
2. Implementation

A 6V battery is used as a power supply, but as the microcontroller works with a 5V input, a 5V DC voltage regulator was used. The servo motor and the distance sensor use 5V as well, while the H-bridge uses the power from the battery directly to control the motors. An external capacitor was connected for noise reduction in the power supply lines purpose.

The first goal of the project was to use an efficient method to control the motors. An L298 H-bridge was used for this purpose. This IC counts with two H-bridges, which work independently one from the other. The H-bridge consist of 2 output pins, which were controlled by 4 inputs pins, 2 of them for the control of the polarity, one for the voltage supply and the last one to enable/disable the outputs. This last pin was controlled by a PWM signal given by the ATMEGA16.

2.1 Program design

Due to the fact that highly accurate timing was not needed, it was necessary just one timer for the generation of the PWM signal for both motors, the execution of the parking algorithm, and for the measuring of the parking lot. Therefore, timeslots every 0.2ms were set up.



In order to generate this PWM signal, an interruption routine activated by a timer was used. This interrupt routine, where all the data was updated in, was programmed to be called every 0.2ms, as established for the timeslots. This was acquired by setting the prescaler to 1 and the OCR0 register to 198. The frequency can be calculated by:

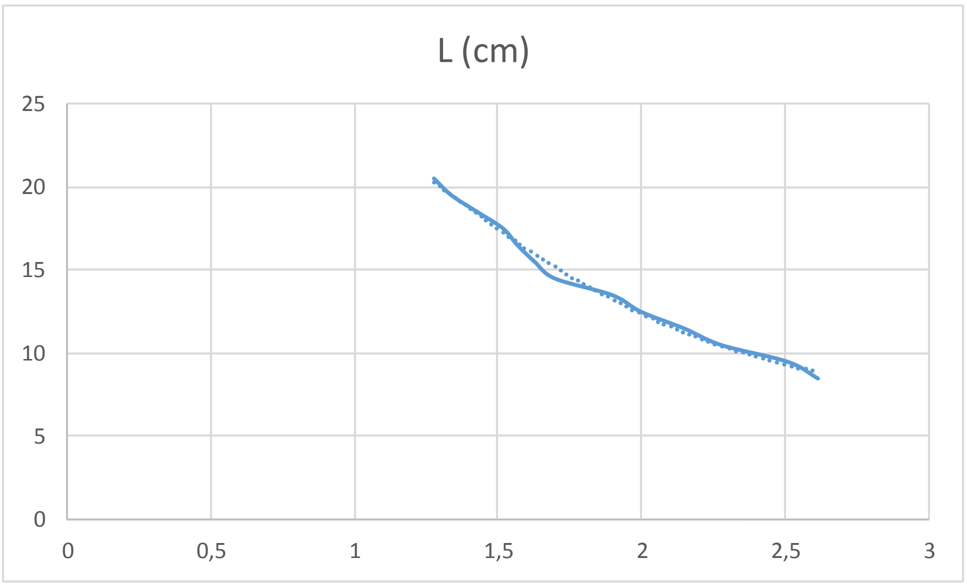
$$f = f_{clk} / N * (1 + OCR0)$$

This will program the ATMEGA16 to count from 0 to the OCR0 value with a frequency of $f = f_{clk} / N$, and every time it reaches the OCR0 value it will enter the interrupt routine and start counting again. This will give a resolution of 1% for the PWM signal, as the period of the signal is set at 20ms. This means that every 20ms the PWM signals are set, and every 0.2ms the interrupt routine has access to this signals to clear them if needed in order to get the desire PWM signal.

The distance sensor was mounted in a servo motor with a 180 degree range. The servo motor needs a PWM signal with a 20ms period, and the turn is handle with the duty cycle (DC) of the signal. For this particular case, another timer is used but with the fast PWM mode. The turn respond to this values:

Position [°]	Duty cycle [ms]
-90	0.6
-45	1
0	1,5
45	2
90	2.3

The SHARP GP2D12 distance sensor used in the project had an effective range of 10 to 80cm, while the perforated breadboard was 16cm long by 8 wide, so even if the sensor distance was placed in the middle of the breadboard, it wouldn't be able to measure with precision the distance needed, so pieces of styrofoam pieces were added. The sensor data sheet shows a distance versus output voltage graph, but when it was tested, the plot did not match the real values, so new points were plotted for different distances, and a new graph was made.



The new graph shows a distance versus output voltage plot, where the trend equation can be approximate as:

$$L = 4.0184 * V^2 – 24.232 * V + 44.762 \text{ (cm)}$$

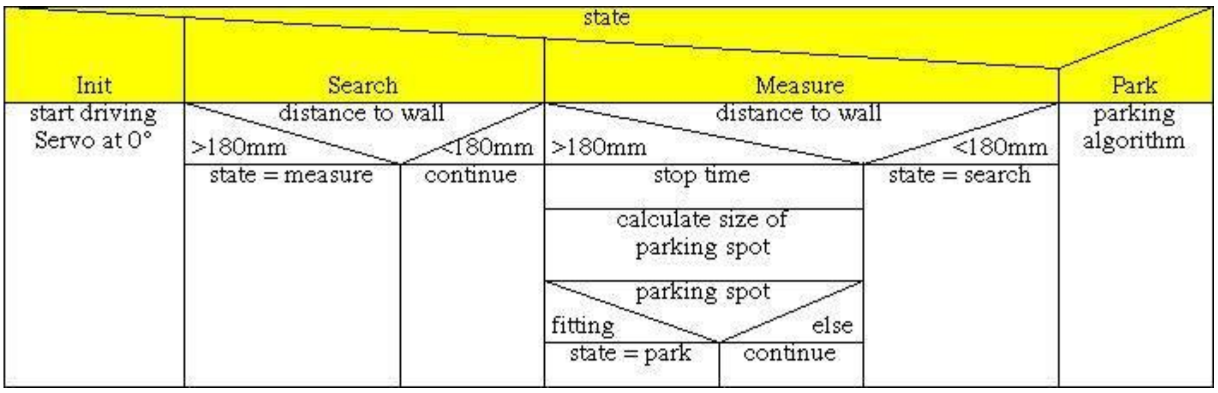
For this purpose, the output voltage of the distance sensor was connected to an Analog-Digital Converter (ADC) port. This will convert the analog input (the output voltage of the distance sensor) in to a 10 bit register with, in this case, an internal 2.56 voltage reference. This mode also needs the connection of an external capacitor from the Vref pin to ground, which was connected. The following equation is used to get the ADC input voltage value:

$$V = ADC * 2.56 / 1024$$

If the V value is replaced in the first equation, the final distance equation will read as:

$$L = 0.25115 * ADC * ADC / 10000 – 0.06058 * ADC + 44.762 \text{ (cm)}$$

2.2 Search and measure parking lot

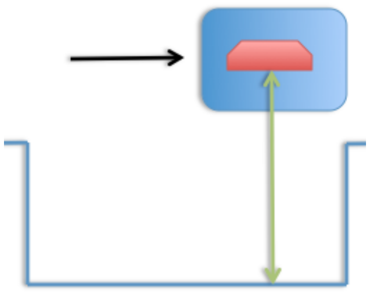


The robot is placed close to the wall (= parking cars) and starts searching for a free parking lot. Therefor it drives close to the wall in a straight line and measures the distance. If there is an empty spot (distance bigger than 180mm) the robot starts calculating the size of the parking lot by driving with a certain speed (20mm/s) and counting the time. If the parking lot is big enough it starts with the parking algorithm if there is a new car before reaching the minimal size of the parking lot it starts searching again.

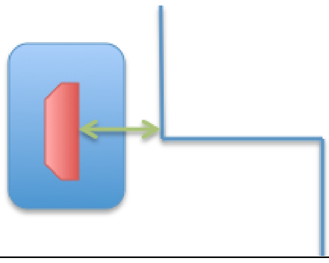
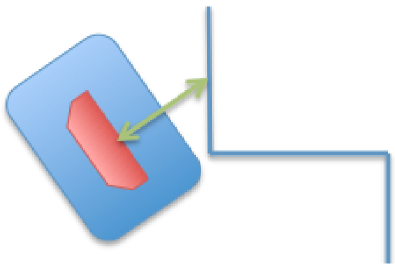
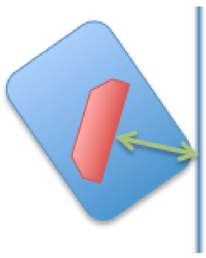
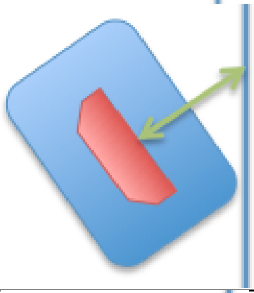
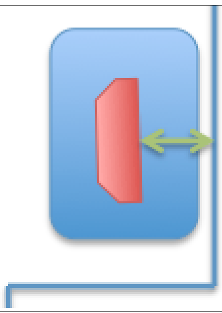
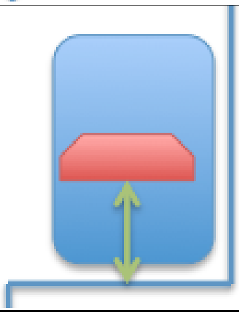
2.3 Parking algorithm

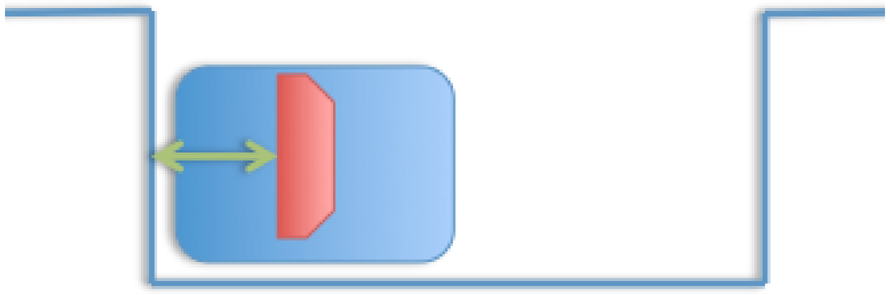
The parking algorithm is programmed as a state machine (Moore automat).

This picture shows the situation at the beginning of the parking algorithm. The robot has already measured the parking lot and decided that it is big enough. Now the parking algorithm will start.



An initial state before every movement is needed to change the driving directions or turn the servo motor. If the servo motor is turned a small waiting loop is added to be sure that the servo motor has enough time to reach its new position.

State	**Description**	**Graphic**
Drive to car	Drive to the end of the parking lot. Stop when reach the wall Save distance to wall	
Init turn left	Change driving direction Calculate distance which will be measured after a 45° turn to the left	
Turn left	Turn left until the calculated distance is reached	
Init Straight	Turn servo motor to the back, left corner of the robot (-45°)	
Straight	Drive straight into the parking lot until the right back corner is close to the wall	
Init turn right	Turn servo motor to the right (0°)	
Turn right	Move forward and turn right at the same time until the distance to the wall isn't getting smaller anymore.	
Init drive to end	Turn servo motor to the back (-90°)	
Drive to end	Drive forward until the robot is close to the wall	



One of the main problem the project had was noise reduction. While trying the self-driving algorithm, the vehicle motors and the servo motor started acting weird for moments. As it seemed noise produced by the motor itself, capacitors were connected to the motor terminals in order to reduce electrical and magnetic noise. After the connections of the capacitors, the motors started working fine.

Another problem experienced during the coding of the algorithm was that the PWM signals generated had some longer pulses every now and then. This was produced because the interrupt routine, which updated the PWM signal data, could not be called if a register in the main program was being updated. For the distance measurement, the equation used required several clock cycles until the new distance value could be calculated, so the equation was split into shorter equations in order to let the timer access the interrupt routine whenever it was ready.

3. Conclusion

Autonomous parallel parking of a vehicle is a feature every person wishes to have in the car. Parking in crowded cities where there are just small spaces has always been an issue. To this day, many manufacturers work in the improvement of this feature, trying to park the vehicles in the smallest space possible. This project presented a basic but efficient solution to this problem, where a model car would search for an empty parking lot using a distance sensor and start parking when it finds one.

Regarding the search of the parking lot, a self-driving algorithm where both PWM signals were the same was implemented. There is no wall following algorithm implemented, and as the road could be irregular, or the direction of the wheels could be pointing anywhere, driving in a straight line should be a priority. This solution could be reached with a PID-Controller, common in automation.

The SHARP GP2D12 distance sensor is not very accurate in terms of proximity, as it has a range starting at 10cm, and the plot given by the datasheet is not always close to the real values. It is recommended the use of a different distance sensor with a closer range.

For the purpose of this project, it was assumed that the parking lot is always in between two cars, so it will start measuring the size of the lot until it reaches again another car, and once with the size of the parking lot, the model car will start the autonomous parking. This is because the closest distance is used as a reference on the turn of the car, so it has to reach another car before parking. Otherwise, it would have to go back to the first car once it knows that the size of the parking slot is big enough. This could be solved with the use of a more accurate distance sensor, so then it could use the reference of the furthest wall.

4. Sources

ATMEGA16 datasheet

SHARP GP2D12 datasheet

L298 H datasheet

Parallax toddler servo mini F/BB datasheet