EEL-5666
Intelligent Machine Design Lab
Final Report

‘Raven’
Mobile Autonomous Robot

Stephen Tobias

Table of Contents:

Abstract:

First I am pleased to say that my 2-legged walking robot works. It can walk and is not completely statically stable. This is a feature I suppose. Using the new 68HC12 proved to be a much better idea than I originally thought, the additional modules proved to be very useful. I did find that the method I was originally planning on using to make the robot walk made it walk poorly. I also theorize on how to make a better walking robot, and why.

Executive Summary:

In the following report I will outline what progress I have made on my robot, what my final conclusions are and what still needs work. I am reluctant to say that I did not complete all of my goals in this project, however I did construct a mobile platform with some basic behaviors that can be build upon and refined to produce a more stable more reliable robot. This version is the first version of this robot and like any program can still be refined. I now have a deeper understanding of how a walking robot works and how to make it move.

Introduction:

It was my intent in this project to design and build a mobile autonomous robot capable of moving about its environment and map its environment as it moves around. To do this I planned to construct a 2 legged walking robot using the new 68HC12 microcontroler. The 2 legged design I thought would be a natural body used in mapping because it has a discrete unit of measure built into it, a single step. Additionally the 68HC12 has additional hardware making it a natural choice for such a complex robot.

Integrated Systems:

The robot has a number of integrated systems to make the completed robot function properly. The robot has a number of servos and IR sensors, the 68HC12 has hardware that can be used to drive these devices.

The servos are controlled by the pulse width modulator and the timer module. Servos require a period between 40 and 60 Hz. The duty cycle of the square wave determines where what position the servo takes. The actual high time for the servos is approximately 1 ms to 2 ms, 1 ms for one side and 2 ms for the other. The square wave can be generated for one by the pulse width modulator very easily by setting the clock divider to 128 and the prescale counter to 3. Which generates a pulse at the rate of 10 kHz. When the period is set to 255, the maximum value the 8-bit period register can hold, the end resulting period is 40 Hz. A value between 2 and 12 hex will move the servo to one of 16 positions. It is a little crude, only having 16 positions, but it can be used for parts of the robot that don't need to be in a large variety of positions. For a larger variety of positions 2 of the PWM registers can be combined together for a larger 16-bit pulse width modulator. This would allow for more diversity. Another solution would be to use the timer module to control the servo. The timer can be used in one of two modes, the input capture mode and the output compare mode, one of the pins can also be used as a pulse accumulator. The output compare can be used to generate a square wave to control the servo with minimal overhead. Additionally the output compare uses the 16-bit timer count register a 16-bit value can be used as the period for the square wave. The resulting duty cycle's value can be much larger than the period used in an 8-bit PWM. The resulting useable range is over a thousand different positions.

The IR sensors are the most straight forward. The hacked sharp IR sensors are connected to the analog to digital converter. When it detects IR it produces a value between ground and 5 volts, which can be read by the analog to digital converter.

Lastly the most important module is the real time interrupt, and the software interrupt. The RTI module is used to switch between tasks every 4 ms. The software interrupt is used simply to stack the registers for when it is deferring tasks to another process.

I also required some additional hardware not standard to the 68HC12. The standard 1 Kbytes of RAM that comes with the 68HC12 is not enough for my code, variable storage and most importantly the map. I expanded the memory to a full 64 Kbytes. This is where most of my problems occurred, like finding memory fast enough for the 8 MHz bus, but the largest problem was caused by a faulty address latch. I also added memory mapped digital inputs and outputs, but later removed them when I was debugging the memory. I also used some additional hardware to slow down the 8 MHz bus clock down to 40 kHz for the IR LEDs.

Mobile Platform:

The mobile platform of the robot is somewhat unique. It is one of very few 2-legged walking robots. The robot has a main body 2 legs with 4 servos on each. Each leg has a hip servo a knee servo and 2 ankle servos. All but one of the servos in a leg are in the same plane to give the leg 3 flexible joints, forewords and backwards. The knees are bent backwards similar to a bird's legs. I thought that that this would prove to be easier to use and provide better weight distribution. The last servo is on the leg was going to be used to turn the robot. Additionally the last servo is used for the tail. This part is used to balance the robot when it is walking or turning.

The robot will actually walk by moving the servos into preprogrammed positions, waiting then moving to a new position until it has completed a step or turn. The draw back is that this causes sudden jerky movement, and causes the robot to not be as stable as I would like it to be. I carefully selected my servos to be special hi torque servos so it could stand on one leg while the other leg could move forward when stepping forward. I even demonstrated that it could stand on one leg, but something happened to it between then and when I started working on the walking system. Instead of creating a statically stable walking robot, I was forced to make it less stable robot, it can't be on one leg except for short times. It now walks by bouncing up moving the leg forward and putting the robot back down into a more stable position, when it had to move a leg forward. Turning also became a problem. I noticed that while it was walking it tended to turn if the leg did not get off the ground completely. When it turns I take advantage of this by not picking up the leg and moving it backward to push it in the direction I want it to turn in. I works, but is extremely messy, and not very stable.

Sensors:

My final robot has only one type of sensor. I had planned on using 2 types of sensors on the robot, IR and bump sensors. The IR sensors were to be used to detect if an object is in the way, and the bump sensors for when the IR sensors don't report the correct data, and used for an auto-calibration. However when I was debugging my memory expansion, I removed the digital input from the memory map. This means that I can't connect the digital switches used for the bump sensors. This means that I must rely on the IR sensors alone to find obstacles. In the long run this probably better since I don't know how I would go backwards when something hits a bump sensor.

Behaviors:

I had initially planned on 3 behaviors, using 5 processes. These behaviors were going to be obstacle avoidance, navigation and mapping. I did not have time to implement the navigation and mapping. The 5 processes are the main process, sensor process, obstacle avoidance process, navigation process and the mapping process. As I said the later 2 were not implemented. The main process is used to move the robot around, and is the only process allowed to manipulate the servos. The sensor process is used to read the sensors, and is the only process allowed to do so. Finally the obstacle avoidance process looks at the sensor data and passes recommendations to the main process.

Experimental Layout and Results:

After many long hours my learned to walk with enough stability to not fall over. The sudden movement causes erratic movement, and some undesired turning, but it can move forward. It moves about 2 to 3 inches per step, and about 15 degrees per turn. It moves by moving the servos to preset positions, the more positions and the less movement between positions the more stable the walking. I don't have a large number of positions so that is why it is unstable. The ultimate solution would be to rewrite the RTI to control the actual servos and use the main to input to the RTI the positions and speed to the servos. The RTI could then take that data and vary the speed of the servos by not giving the timer the final position, but incrementing the period slowly to produce a more stable movement.

Conclusion:

I still stand by my opinion of that a walking robot makes a better platform for a mapping or exploration robot. However the method I am using to make it walk is not a very effective. The sudden movements make it turn as it moves and makes it nearly fall over. I have also learned that the turning algorithm is insufficient and too inaccurate for mapping. The walking dose give me a more accurate reading of distance, but the poor turning is much worse then a more standard robot with wheels. If I smooth the movements and refine the turning it might make a better mobile platform.

The 68HC12 that I am using to control the robot is the very best decision I could have made. The additional hardware, namely the PWM made the servo controller almost trivial. The 68HC12 also has more timer ports that can be used as either input capture or output compare, meaning that I have up to 8 timer ports to control 8 servos, unlike the 69HC11 which only had 4.

In the future I would like to debug and smooth out the walking so it no longer looks like is about to fall over, and find a more accurate way to turn so I could finally implement the mapping navigation behaviors. In all this was definitely a worth while project to undertake.