Proc. 2002 Australasian Conference on Robotics and Automation
Auckland, 27-29 November 2002

Mechatronics for Designing Intelligent Machines

W.L. Xu
Institute of Technology and Engineering
Massey University
Palmerston North, New Zealand
W.L.Xu@Massey.ac.nz

G. Bright
Institute of Technology and Engineering
Massey University
Auckland, New Zealand
G.Bright@Massey.ac.nz

Abstract

This paper shows how a class of undergraduate students at Massey conducted an open-ended, less specified mechatronics project, and to what extent they could integrate knowledge and skills across various subjects into the project. The intelligent robot PKBot introduced in this paper was one of the designs for the theme Robotic Lawnmower in 2001. The project was roughly specified in the following: The robot must simulate lawn mowing, by installing markers on its body, which can map where it has been on the surface provided. The robot must be able to distinguish any obstacles set up on the surface and navigate around them, without any outside interference. The entire surface provided to simulate the lawn mowing process on must be totally covered by the robot, or as much of it as possible as the case may be. The robot should navigate the surface area as quickly as possible.

1 Introduction

Each year at Massey University, New Zealand a class of 4th-year BE Automation and Control students taking Mechatronics course are asked to design a small machine with embedded microcontroller, sensors, and intelligence. The design project is carried out in groups of 4 students, 3 hours a week for 12 weeks. It takes 40% of the final course assessment. It is an open-ended, team based and less specified project, and its theme changes from year to year. The main objective of this mechatronics design project is to develop students' competence in applying knowledge and technologies they have learnt in mechanical, electrical and electronic engineering and computer technology (in particular, in motion actuation, sensing and signal processing, and microprocessor control), and to encourage them to practise their systematic and open-minded thinking for solving real problems as a team. The course, which is taught 3 hours per week for 12 weeks, covers mainly mechatronics design philosophy, motion sensing and actuation, microcomputer control and real-time interfacing, intelligent control and case studies.

This paper shows how a class of undergraduate students at Massey conducted an open-ended, less specified mechatronics project, and to what extent they could integrate knowledge and skills across various subjects into the project. Through this open-ended, team-based, less defined project, students should also demonstrate their problem solving abilities and team working/cooperating spirits. The intelligent robot PKBot [1] introduced in this paper was one of the designs for the theme Robotic Lawnmower in 2001

2 Mobile Platform

After a few conceptual designs were evaluated, PKBot went on the drawing board of the SolidWorks [2] program. By constructing 3D model of the major components it was possible to design a compact yet versatile robot. PKBOT was based on a circular design utilising differential drive system for added manoeuvrability.

The designed robot had a two-part chassis, a lower and upper plate, which offers several benefits. Firstly this design was structurally necessary to support all the components of the robot. Secondly it was thought that there might be an undesirable noise level emanating from the motors, therefore it was necessary to separate the circuitry from the motors and battery by way of the top plate. Caster wheels with a 30° rake, while adding stability to the robot also allowed it to rotate freely on the spot making it very agile in the field. The base was composed of two stainless steel plates 160mm in diameter, these were held apart by four aluminium rods about 55mm in length, as shown in Figures 1 and 12. Two stepper motors were mounted to the lower plate via existing holes in the motor housing.

3 Mobile Platform

The strategy that was developed to ensure the ‘lawn-mower’ covered the entire area requires the stepper motors to be controlled depending on a set path. This path is determined by the state of the sensors at particular times (refer to Section 6 Navigational and Obstacle Avoidance

Strategy). To implement the strategy, a micro-controller of some sort is necessary in order to store specific instructions for particular situations. The sensor output (high or low) will be directly linked to particular pins and resultant signals will be computed and sent to the motor control circuitry; thus driving the stepper motors in the desired fashion.

A Winbond W78E51B 8-bit micro-controller was used as the brain of PKBot [3]. All that was needed to implement the micro-controller was input/output pins attached to the micro, a crystal oscillator circuit, and a regulated power supply. Once the circuit was designed the next step was to design a Printed Circuit Board. To do this, a program call Winboard PCB [4] was used to lay out the tracks of the circuit board, taking care not lay tracks too closely or to overlap them. Shown in Figure 2 is the completed micro-controller PCB for PKBot. The micro-controller was programmed in C language to implement the navigational and obstacle avoidance strategy given in Section 6. A program called ProView32 [5] was used to allow the C program developed to be compiled to HEX format. This was then downloaded to the chip using a software package SuperproZ. [6]

4 Actuation and Motor Controller

PKBot was made mobile via two 3.5V, 0.35A, 200S/R Stepper Motors. For the Micro-Controller to impart motion on these motors, a motor drive circuit is required. The SGS-Thomson L297/L289 Stepper Controller/Driver [7] combination was implemented. The advantage of using this combination of semiconductors is that it greatly simplifies the operation of the robot. For a stepper motor to operate, coils within the motor need to be energised in a particular order to generate the stepping action of a step motor. Generally this can be taken care of by implementing a step table in the Micro-Controller. The minimum input signals required to operate a step motor is a variable clock signal (speed control), and direction signal (CW/CCW). Following the data sheet for L297/L298, a motor control PCB was fabricated, as shown in Figure 3.

5 Sensors

The interface between the robot and the environment are six identical infrared sensors that surround the robot (refer to Figure 12). The range of the sensors is approximately 50 mm. The sensors were made up of an infrared Light Emitting Photodiode and the IS471F integrated circuit [8]. The IS471F is a four-pinned device that provides a means of detection for Infrared Light as well as providing a square wave at approximately 38 kHz to pulse the Infrared LED.

Two range sensors (Sharp GP2D02 Infrared Ranger [9]) were used to keep PKBot perpendicular to walls and objects by comparing the two distances and positioning the robot relative to these. The output of the range sensor is either high or low at each instance when the clock signal switches from a high state to a low state. These values are captured in an 8-bit array, the first number representing the MSB and the last, the LSB. The pin is then switched high for 2ms, then low for 1.5ms. The 8-bit array must then be converted to a decimal value between 0 and 255. To accurately convert this to a distance, a non-linear mathematical function was discovered experimentally that converts the number to a distance.

6 Navigational and Obstacle Avoidance Strategy

The basic strategy that was implemented is to start in a corner, with the side wall on the robot's left, and simply drive backwards and forwards, traversing the length of the field until the robot ends in one of the opposite corners. If an object is encountered along the way, the robot would navigate around it and, as quickly as possible, get back to its default action of traversing the field. This strategy is explained in greater depth below.

6.1 Searching for Corner

In a general situation, a fully automated lawn-mower would be placed anywhere on the lawn and switched on and it would then proceed to cut the lawn until finished without any human interaction. Taking this into account, a section of code was constructed to position the robot in a corner of the lawn where the sidewall was at its left, illustrated below in Figure 4.

The robot is dropped (at random) into the field. PKBot then goes forward until it senses an object in front of it. PKBot then proceeds to position itself so that it is parallel to the object by use of the range sensors (positioned on the back, pointing behind robot). PKBot will then turn 90° clockwise (CW) and check if a wall is at its left, and behind it. If so it is in a corner and can proceed to the next stage, if not it will go froward until it hits another object and repeat the above process until it is in a corner.

6.2 Travelling the field

Once the robot is in a corner with a wall at its left, it begins traversing the field. It begins by setting the DIRECTION variable to 0 and moving forward, step by step, until it hits the opposite wall. Each step it takes, a variable DISANCE is incremented. Once it hits the end for the first time, the variable BOX_WIDTH is set to the value stored in DISTANCE. It then uses the front sensors to align itself perpendicular to the wall. If one of the front sensors is activated and the other isn't, then PKBot rotates on the spot until both are. That is,

- If DIRECTION = 0, PKBot then turns 90° CW, moves forward 350 steps (represents the width of the robot), and then turns 90° CW again. DISTANCE is reset to 0, and DIRECTION is set to 1.
- Otherwise (DIRECTION = 1), PKBot turns 90° CCW, moves forward 350 steps (represents the width of the robot), and then turns 90° CCW again. DISTANCE is reset to 0, and DIRECTION is set to 0.

PKBot then moves forward again (incrementing DISTANCE each step again) until it hits an object. If DISTANCE is not equal to BOX_WIDTH (within a set

error margin) then PKBot has hit an object and goes into object avoidance mode. Otherwise it continues with the traversing code. This strategy is presented in Figure 5.

6.3 Obstacle Avoidance

It is assumed that all the objects are rectangular in shape and are wider than the distance between the front sensors (~14cm). Extra sensors could be added on the front to decrease the minimum width of the objects.

(1) Square Object (Figure 6)
When a front sensor (F_SENS_1 or F_SENS_2) is activated and the DISTANCE counter is less than BOX_WIDTH, PKBot realises that it has encountered an object and must navigate around it. It does this by firstly turning CW 90°, then moving forward until the back left sensor (L_SENS_2) does not sense anything. The number of steps it travelled is recorded as OFFSET. It then proceeds to turn 90° CCW and moves forward a set distance so that L_SENS_2 is past the edge of the box. It then continues to go forward until L_SENS_2 is past the object, incrementing DISTANCE for each step taken. PKBot then turns 90° CCW again, moves forward the distance OFFSET and then turns 90° CW.

(2) Wall Object
There are three situations that need to be accounted for concerning objects positioned against a wall. These are illustrated below:

(A) During a traverse (part 1)
When PKBot hits an object during a traverse it always treats it as a square object and precedes to travel around it. When the front sensors are activated while travelling in the x direction towards the wall, it realises then that it is an object against the wall (Figure 7). If OFFSET is less than the standard 350 then it turns 90° CW and moves forwards 350-OFFSET steps and does another 90° turn CW. DISTANCE is set to 0 and DIRECTION is set 1 if it was 0, or 0 if it was 1. Then returns to the traversing code.

(B) During a traverse (part 2)
This is similar to the above method until it realises that OFFSET is greater than 350. It then does a 180° turn, moves forward WALL_OBJECT steps, turns 90° CCW. Then PKBot moves forward OFFSET-350 steps and turns CW 90°. DIRECTION is set to the opposite of what it was again, and DISTANCE is set to the value stored in WALL_OBJECT. The avoidance strategy is presented in Figure 8.

(C) During a U-turn
If the front sensors are activated when PKBot is doing a U-turn in the traversing stage, it has either reached the end, or has hit an object (Figure 9). It overcomes this by turning 90° CW and moving forward until L_SENS_2 is past the object, the distance it travelled is stored in WALL_OBJECT. PKBot then turns 90° CCW and moves forward for OFFSET-350 steps, then turns 90° CW. DIRECTION is set to its opposite value and DISTANCE is set to WALL_OBJECT.

6.4 Detecting the End

While the robot is avoiding a wall object and the front sensors are activated before L_SENS_2 is clear of the object, then the 'object' is actually the end wall and the robot is in the corner, and the code activates an infinite while loop which basically halts the robot indefinitely.

7 Summary

Tests and runs showed that PKBot (as pictured in Figure 10) was very stable in effectively filling the pre-specified field while navigating around objects randomly placed. The design of PKBot for mecahtronics project met the objectives defined successfully. A video clips of the robot running is available at the website [9].

The key technologies that the group of 4 students for PKBot applied in designing and building PKBot are CAD for mechanical design, motion sensing, motor control, interfacing circuitry and PCB making, microprocessor and its programming, and intelligent behavior control. It should be noted that the intelligence for navigation and avoidance implemented is fairly sophisticated in terms of a undergraduate project.

Given a set of very generic project definitions, the final robots that could be expected from different design groups are of course different. Other designs using different sets of microcontroller, sensors and motion actuation and intelligence are available also at the above website.

Acknowledgements

PKBot in this paper was designed and built by Tane Coe, Alistair Cook, Darryl Wai, Michael Young who were 4th-year BE Automation and Control students. Mr. K. Mercer, the electronic technician of the Institute of Information Science and Technology, Massey University, provided technical support throughout entire project.

References

1. T. Coe, A. Cook, D.Wai, M. Young, "The Final Report of PKBot", Institute of Technology and Engineering, Massey University, June, 2001
2. http://www.solidworks.com
3. http://www.winbond-usa.com
4. http://www.ivex.com/sales/products/winboard/index.shtml
5. http://www.fsinc.com/devtools/products/DS-PV32.htm
6. http://www.generaldevice.com/products/superpro_z__40pin_device_program.htm
7. http://www.acroname.com/robotics/parts/R132-IS471F.html
8. http://www.acroname.com/robotics /parts/R19-IR02.html
9. http://www.acroname.com/robotics/parts/R19-IR02.html
10. www.massey.ac.nz/~wlxu

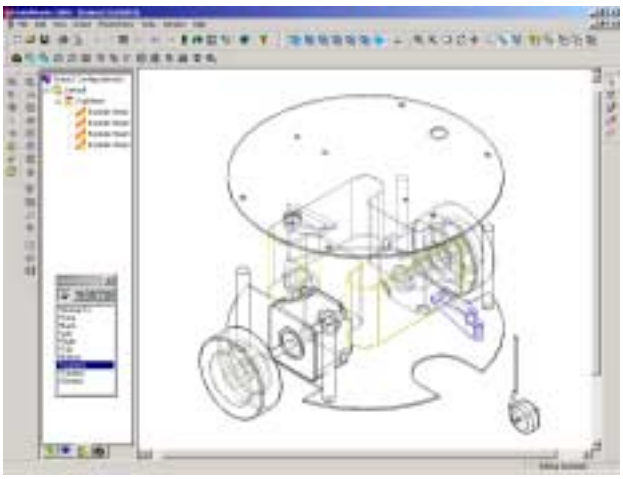

Figure 1 SolidWorks Design and 3D Construction

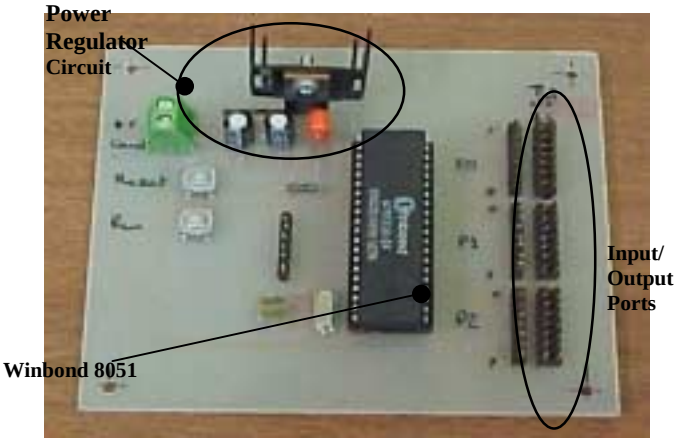

Figure 2 Completed Micro-Controller PCB

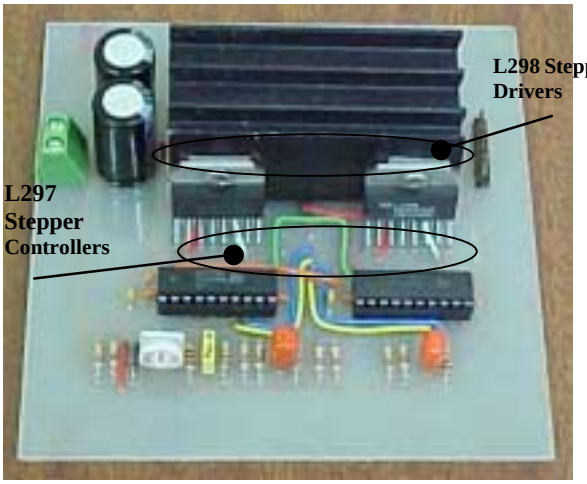

Figure 3 Completed Motor Controller PCB

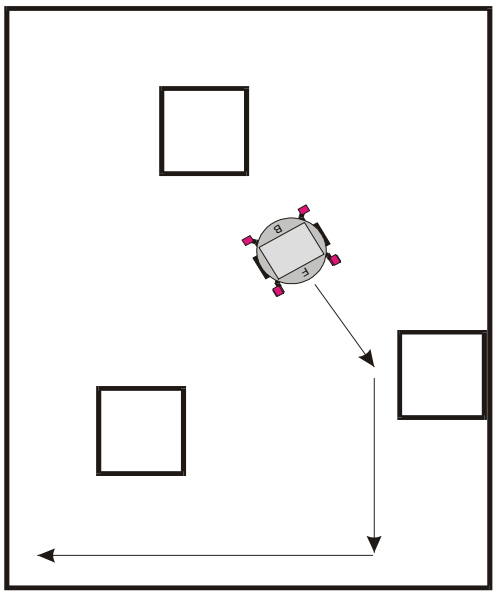

Figure 4 Where to Start

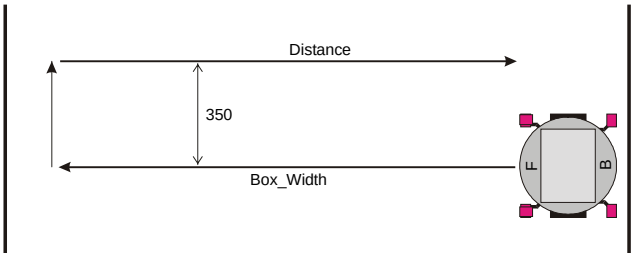

Figure 5 Traversing the Field

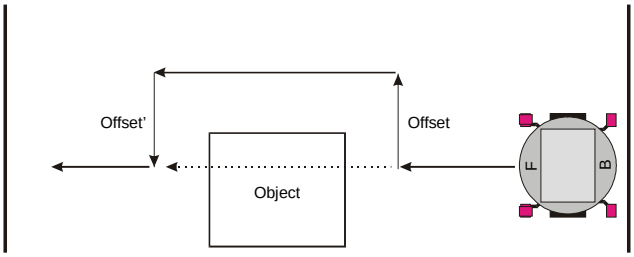

Figure 6 Avoiding an Object

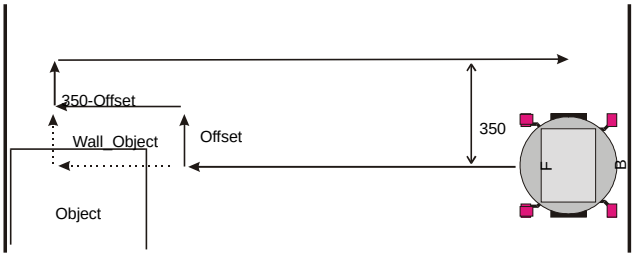

Figure 7 Avoiding an Object by Wall - During a Traverse (part 1)

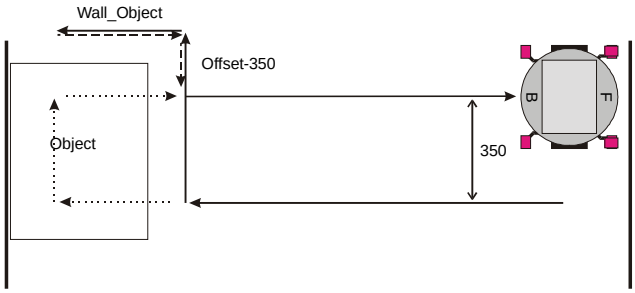

Figure 8 Avoiding an Object by Wall - During a Traverse (part 2)

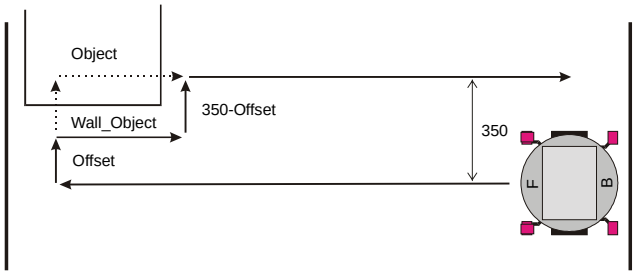

Figure 9 Avoiding an Object by Wall - During a U-turn

(a)

(b)

Figure 10 PKBot, from Concept to Reality, and Its Navigation