Far East Journal of Electronics and Communications
Volume 3, Number 2, 2009, Pages 141-153
Published Online: September 14, 2009
This paper is available online at http://www.pphmj.com

DEVELOPMENT OF MOBILE ROBOT WITH AN INTER-INTEGRATED CIRCUIT SYSTEM

P. SURACHAI

Faculty of Engineering
Srinakharinwirot University
Thailand
e-mail: surachap@swu.ac.th

Abstract

This paper explains a new way of communication between mobile robots based on PC-Card and sensors by using I^2C system. The hardware interface with the Inter-Integrated Circuit System or shortly I^2C bus system is mainly described here. The mobile robot components, I^2C bus system and I^2C device are also introduced. The software is developed to control sensor data from the robot (master device) to slave devices based on I^2C bus with C++. Finally, the performance of I^2C bus system is compared with the previous system by experiment. The robot runs with localization algorithm in square shape. Localization algorithm is developed with the concept of Markov to update robot's position by sensor readings from the environment.

I. Introduction

The environment perception of the robot can be accomplished by means of sensors. The sensor data processing can be used by connecting on the robot by many ways. Nowadays, microcontrollers play the important role because they can be found easily in the electronic shop, and widely developed in many companies, but the

Keywords and phrases: I^2C, compass, mobile robot.

Received February 6, 2009

Inter-Integrated Circuit (I^2C) bus is recently introduced. The I^2C bus is an industry standard synchronous serial data communications bus used by many common consumer electronics and embedded systems components. Originally developed by Philips for television tuner integrated circuits, I^2C bus interfaces are now found on hundreds of consumer electronics chips, including data converters, EEPROMs, thermal sensors and real-time clocks [5]. Since I^2C is a relatively low bandwidth communications bus, it has found widespread to use in control interfaces in signal processing devices with separate data interfaces. Typically, I^2C bus analysis is done with expensive test and measurement equipment, dedicated microcontroller based system or desktop computer with a dedicated interface card. This paper describes an I^2C bus system implemented on the Pioneer-II. Section II explains the major features of Pioneer-II robot. In Section III, concepts of the I^2C bus are briefly reviewed. Section IV describes the organization of the hardware interface between the Pioneer-II robot (as master device) and slave device and overview of the project is given. In Section V, software is developed under Visual C++. Finally, Section VI contains the conclusions and ideas for future work.

II. The Robot Architecture

In this section, the global hardware architecture is detailed and each module is presented. It consists of two parts, the robot and the addition system (I^2C bus system). For the first part, our laboratory has a Pioneer-II [1, 2] as shown in Figure 1, a family of mobile robots. They are small and intelligent robots. These are truly off-the-shelf, “plug and play” mobile robots, containing all of the basic components for sensing and navigation in a real world environment, including battery power, drive motors and wheels, position/speed encoders. They are all managed via an onboard microcontroller and mobile robot server software. For addition system, the I^2C bus is recommended that can integrate more sensors on the robot. This system will be explained in the next section.

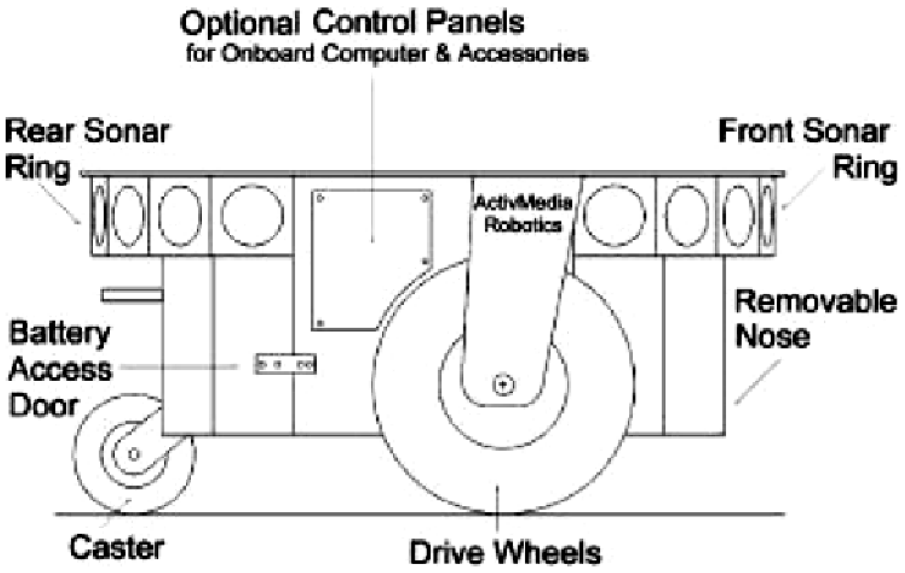

Figure 1. Overall of the Pioneer-II architecture.

A. Main components of Pioneer II

All Pioneer-II models use a high-performance 20MHz Siemens 88C166 based microcontroller with independent motor/power and sonar-controller boards for a versatile operating environment. The controller has two RS232-Standard communications ports and an expansion bus to support many accessories available for Pioneer as well as our own custom attachments. Also, the Pioneer-II comes with high precision (9,850 ticks-per-revolution) wheel-motor encoders for finer odometry, and translation and rotation speed controls. The Pioneer-II also supports a full complement of sixteen sonars (eight fronts and eight rears) for nearly seamless object detection.

B. Microcontroller

Pioneer-II's microcontroller uses a 20MHz Siemens 88C166 microprocessor with integrated 32K flash-ROM. The microcontroller also has 32K of dynamic RAM, two RS232-compatible serial ports, several digital and analog-to-digital, and PSU I/O user-accessible ports, and an eight-bit expansion bus. All the I/O ports, except those used for the motors, encoders and sonar are available to the user for Pioneer-II accessory hardware, which we may control through the P2OS.

C. Motors and position encoders

Pioneer-II's drive system uses high-speed, high-torque, reversible DC motors. Each front drive motor includes a high-resolution optical shaft encoder that provides 9,850 ticks per wheel revolution (19 ticks per millimeter) for precise position and speed sensing and advanced dead-reckoning.

III. The I^2C Bus System Overview

When connecting multiple devices to a microcontroller, the address and data lines of each device were conventionally connected individually. This would take up precious pins on the microcontroller, result in a lot of traces on the PCB (Print Circuit Board), and require more components to connect everything together. This made these systems expensive to produce and susceptible to interference and noise. To solve this problem, Philips developed I^2C bus in the 1980s. I^2C bus is a low-bandwidth, short distance protocol for on board communications. The name I^2C is shorthand for a standard Inter-IC (integrated circuit) bus. I^2C bus provides good support for communication with various slow, on-board peripheral devices that are accessed intermittently, while being extremely modest in its hardware resource needs. It is a simple, low-bandwidth, short-distance protocol. Most available I^2C bus devices operate at speeds up to 400Kbps, with some venturing up into the low megahertz range. The I^2C bus system is easy to use to link multiple devices together since it has a built-in addressing scheme.

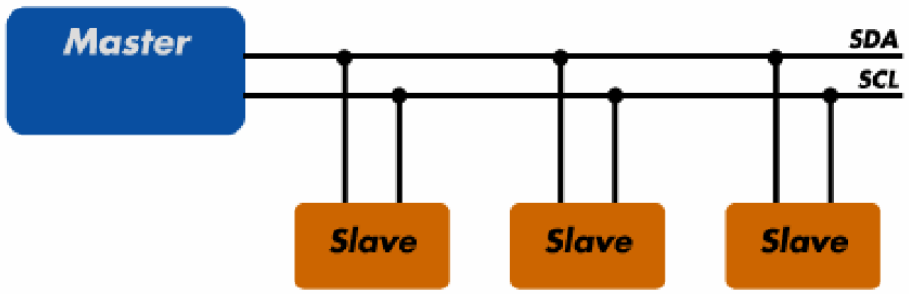

Figure 2. I^2C bus has only two lines in total.

The I^2C bus is a two-wire serial bus as shown in Figure 2. There is no need for chip select or arbitration logic, making it cheap and simple to implement in hardware. The two I^2C signals are serial data (SDA) and serial clock (SCL). Together, these signals make it possible to support serial transmission of 8-bit bytes of data-7-bit device addresses plus control bits-over the two-wire serial bus. The device that initiates a transaction on the I^2C bus is termed the master. The master normally controls the clock signal. A device being addressed by the master is called a *slave*. In a bind, an I^2C bus slave can hold off the master in the middle of a transaction using what is called *clock stretching* (the slave keeps SCL pulled low until it is ready to continue). Most I^2C bus slave devices do not use this feature but every master should support it. The I^2C bus protocol supports multiple masters but most system designs include only one. There may be one or more slaves on the bus. Both masters and slaves can receive and transmit data bytes. Each I^2C bus compatible hardware slave device comes with a predefined device address, the lower bits of which may be configurable at the board level. The master transmits the device address of the intended slave at the beginning of every transaction. Each slave is responsible for monitoring the bus and responding only to its own address. This addressing scheme limits the number of identical slave devices that can exist on an I^2C bus without contention, with the limit set by the number of user-configurable address bits (typically two bits, allowing up to four identical devices). In Figure 3, the master begins to communicate by issuing the start condition. The master continues by sending a unique 7-bit slave device address, with the most significant bit (MSB) first. The eighth bit after the start, read/not-write, specifies whether the slave is now to receive or to transmit.

Figure 3. The I^2C bus communication.

This is followed by an ACK bit issued by the receiver, acknowledging receipt of the previous byte. Then the transmitter (slave or master, as indicated by the bit) transmits a byte of data starting with the MSB. At the end of the byte, the receiver (whether master or slave) issues a new ACK bit. This 9-bit pattern is repeated if more bytes need to be transmitted. In a write transaction (slave receiving), when the master is done transmitting all of the data bytes it wants to send, it monitors the last ACK and then issues the stop condition. In a read transaction (slave transmitting), the master does not acknowledge the final byte it receives. This tells the slave that its transmission is done. The master then issues the stop condition.

IV. Hardware Interface between Robot and I^2C Bus Devices

In this section, the hardware interfaces between I^2C bus system and sensors based on I^2C bus via serial port of PC-Card on Pioneer II, robot at institute as in Figure 4. In order that the robot can receive data from sensors, the I^2C bus system is introduced. Because the robot has only one free serial port, it will be very hard to connect more sensors. In the previous work, microcontroller and electrical circuits are used to receive data from sensors and then send the sensor data to the robot. But it has some problems that microcontroller and electrical circuits cannot be conveniently developed if new sensors must be integrated. Because microcontroller program must be deleted and then for new sensor support reprogrammed. Furthermore, new electrical circuits must also be required for the new sensor integration. In order that the signal line of serial port from the robot can connect to devices on I^2C bus, the electrical master module must be designed to produce signal SDA and SCL. For our work, we select the electrical board from Inex-Company to produce signal and the board works as master device. This board is connected to the PC-Card in the robot through serial port. It means that the robot now works as master device to produce signal. The I^2C slave device is compass sensor module as shown in Figure 5. This sensor can work on the I^2C bus without addition circuit. This compass module has been specifically designed for the use in robots as an aid to navigation. The aim was to produce a unique number to represent the direction the robot is facing. The compass uses the Philips KMZ51 magnetic field sensor, which is sensitive enough to detect the Earth's magnetic field. The output from two of them mounted at right angles to each other is used to compute the direction of the horizontal component of the Earth's magnetic field.

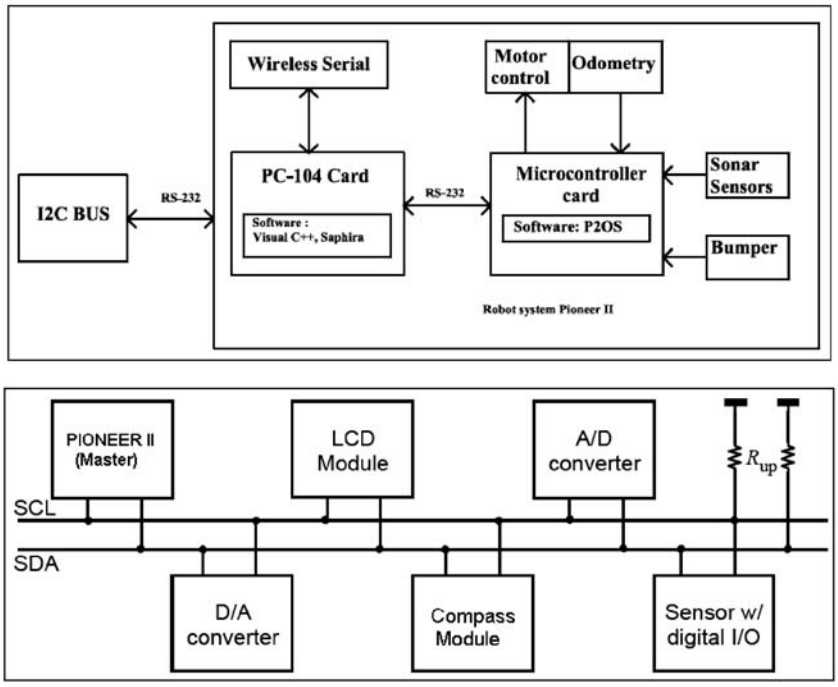

Figure 4. Hardware interface between the robot system and I^2C bus.

Figure 5. The compass module slave device.

The compass module requires a 5v power supply at a nominal 15mA. The pulse width varies from 1mS (0°) to 36.99mS (359.9°) - in other words, 100uS/° with a +1mS offset. On I^2C bus, there is an important consideration that consists of the address from manufacturer and the address from user.

V. Software Development

In the previous section, we described the communication between the Pioneer-II robot (worked as master device) and sensor module (worked as slave device). Now we describe step by step that the robot can control SCL and SDA lines to get sensor data from slave module. The software is designed to control data between the robot and sensor device, which is programmed based on Visual C++. To control data line on the I^2C bus, the step ordering of function based on Visual C++ must be carefully accurately considered and programmed, because if one step misses or does not complete, all devices on this bus will fail. The main function used for programming will be now detailed. All conditions are only generated by robot (as master device). The two main functions are I2C_START ( ) and I2C_STOP ( ). The I2C_START ( ) function produces the START condition as shown in Figure 6. This condition is a HIGH to LOW transition on the SDA line while SCL is HIGH. And the I2C_STOP ( ) will produce the STOP condition. This procedure is a LOW to HIGH transition on the SDA line while SCL is HIGH. Before the START condition will begin and after STOP condition finished, the bus is considered to be always free condition, both the lines must be HIGH. The next main function is I2C_ACK ( ). Figure 7 shows that after the robot (master device) sent data to slave device finished, the slave device must send back the acknowledgement that the slave device received data already. Figure 8 shows a complete transfer cycle associated with a frame of data. First, the master initiates a write by asserting logic-0 at bit-8, where a slave address is defined by the other 7-bits. An acknowledge signal then follows from the slave as specified in bit-9. The second and third bytes are the data and acknowledge signal. The 7-bits addressing allows 127 devices on the I^2C bus, by using 2-bytes address, which can be extended further. The last two main functions are to control and get data from slave device. It consists of I2C_SEND ( ) and I2C_RECEIVE ( ) functions. This I2C_SEND ( ) function is sent always by robot (master device) to control and set slave devices configuration and the I2C_RECEIVE ( ) function is

also sent by robot to receive data from slave devices. Now we show the example function ordering to connect a slave device through I^2C bus to receive data. All step functions are programmed based on Visual C^{++} and detailed below:

Step 1 : I2C_START ( ),

Step 2 : I2C_SEND ( ) - Call slave device address from robot and write mode configuration,

Step 3 : I2C_ACK ( ),

Step 4 : I2C_SEND ( ) - Enable analog output, single mode and analog channel selection,

Step 5 : I2C_ACK ( ),

Step 6 : I2C_STOP ( ),

Step 7 : I2C_START ( ),

Step 8 : I2C_SEND ( ) - Read mode; start to read data from analog input of selected channel,

Step 9 : I2C_ACK ( ),

Step 10 : I2C_RECEIVE ( ),

Step 11 : I2C_ACK ( ),

Step 12 : I2C_STOP ( ).

With these steps, the robot can get data from only one module for one time. If it wants to get data from this module once again or from another module, the robot must rerun once again from Step 1 to Step 12.

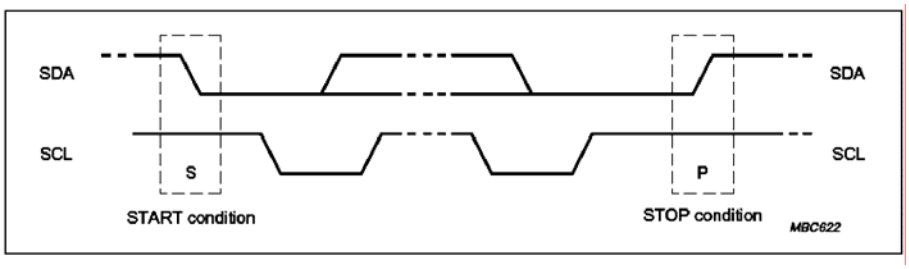

Figure 6. Start and stop conditions of two lines.

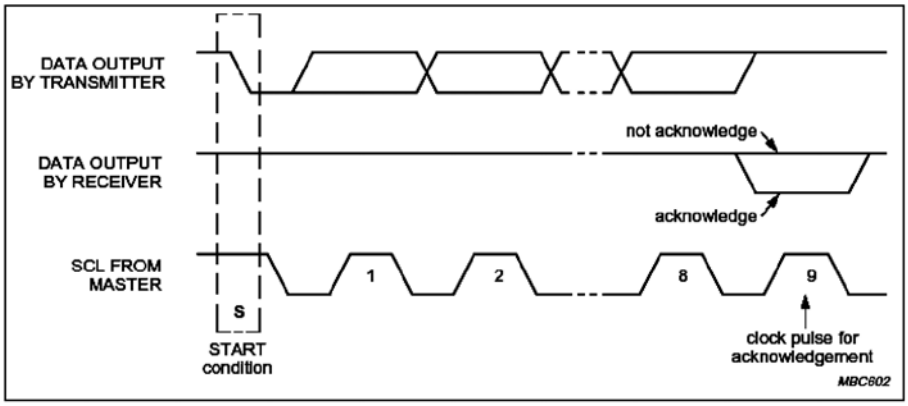

Figure 7. Acknowledge condition.

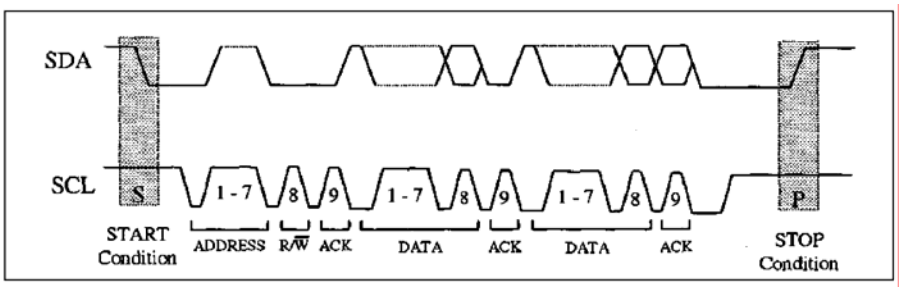

Figure 8. Data transfer on the I^2C bus.

VI. Experiment

From this section, we develop the concept of Markov localization to update robot's position by sensor readings from the environment. We compare performance sensor readings from environment between the previous system and the new system. In the previous system, the robot read sensor data by using microcontroller and now we develop system by using I^2C bus. In Markov localization, progressing from $X_{n\backslash n}$ to $X_{n+1\backslash n+1}$ proceeds in two steps:

Prediction step

$$X_{n\backslash n} \rightarrow X_{n+1\backslash n},$$

Update step

$$X_{n+1\backslash n} \rightarrow X_{n+1\backslash n+1}.$$

Notation. Let 0, 1, 2, ..., k be a sequence of time steps. $X_{i\backslash j}$ is the estimated orientation of the robot at time i and sensor information at time j. By using this notation, the estimated position of the robot from dead-reckoning is $X_{n\backslash 0}$, that is, no sensor information is utilized. The effect of sensor reading in the update step is dependent only on the current orientation of the robot. We make the experiment that the robot runs ten rounds in square shape $4\text{m} \times 4\text{m}$ and record the robot's orientation information from compass sensor. The robot runs both clockwise (CW) and counter clockwise (CCW) in square shape (90 degree at corner). We measure robot's orientation only at four corners with measure tape and record robot's orientation error in the following tables.

Table 1. Sensor reading by using previous system

Corner	Orientation error from 90 degree in CW (degree)	Orientation error from 90 degree in CCW (degree)
1	4.53	4.35
2	3.22	5.76
3	3.56	3.43
4	3.67	4.89

Table 2. Sensor reading by using I^2C system

Corner	Orientation error from 90 degree in CW (degree)	Orientation error from 90 degree in CCW (degree)
1	2.12	1.34
2	1.34	2.13
3	2.11	1.22
4	1.43	1.88

For previous system read sensor data by using microcontroller and interface circuit.

VII. Conclusions

The main purpose of this paper has been to introduce a new way of communication between mobile robot and sensors. The I^2C bus system is considered to use in our system, because it can be conveniently developed and modify our system if new sensors must be integrated or more analog sensors are used. The I^2C bus system consists of master and slave devices that the master device is integrated on the Pioneer-II robot and compass module work as slave devices. The integrated circuits or slave devices support I^2C bus system. It works only with two signal lines and has independently electrical module of each other. Considerately, the device address must be only correctly specified that the company has own configuration for each device. For software developing, the software is developed to communicate the robot and compass module by means of I^2C bus system. The developed software is programmed in function ordering. All functions must be programmed in the I^2C bus format to control SDA and SCL lines in order that the robot as master device can receive sensor data from slave device. This software is programmed by Visual C++ and combined with Saphira/Aria library from Active Media Company. Finally, the robot is tested by experiment that it runs with Markov localization algorithm in square shape both clockwise and counter clockwise. The I^2C bus system can improve the accuracy of robot's orientation 50% more compared with the previous system.

References

[1] ActivMedia, Saphira Software Manual, ActivMedia Robotics LLC, 44 Concord Street, Peterborough, NH 03458, USA, 1997.

[2] ActivMedia, Pioneer 2/PeopleBot Operations Manual, 8. Aufl, ActivMedia Robotics LLC, 44 Concord Street, Peterborough, NH 03458, USA, 2001.

[3] Ph. Bonnifait and G. Garcia, Design and experimental validation of an odometric and goniometric localization system for outdoor robot vehicles, IEEE Trans. Robotics and Automation 14 (1998), 541-548.

[4] J. M. Flynn, Understanding and using the I^2C bus, Embedded Systems Programming, 1997.

[5] D. Kalinsky and R. Kalinsky, Introduction to I^2C, Embedded Systems Programming, 2001.

[6] L. Kleeman, Optimal estimation of position and heading for mobile robots using ultrasonic beacons and dead-reckoning, Proceedings of the IEEE Int. Conference on Robotics and Automation, 1992, pp. 2582-2587.

[7] Philips Semiconductor, I^2C Bus Specification, version 2.1, January 2000.

[8] S. Sarns and J. Woehr, Exploring I^2C, Embedded Systems Programming, 1991.

[9] Dr. Susanne Wigard, Visual C^{++} 6, Das bhv Taschenbuch.