Design a Mobile Robot Operating on Linux Operational System for Educational Purposes

Thuyen Van NGO

University of Technical Education Ho Chi Minh City, Vietnam

thuyen.ngo@hcmute.edu.vn

Dong Hung NGUYEN

Vietnam National University Integrated Circuit Design Research and Education Center, Ho Chi Minh City, Vietnam

nhdong@icdrec.edu.vn

ABSTRACT

This paper presents the design and implementation of a mobile robot which can be controlled on Linux using Player-like library with client/server mechanism. The user program acts as a client which interfaces with the motor and sensors via a server. The robot is differential driven and equipped with a ring of twelve sonar sensors, two encoders and a camera. Robot can send a command to control the robot on the open source code, named Player. The robot was equipped with a ring of 12 ultrasonic sensors to provide the information about the obstacles and a digital compass, encoders to provide its orientation in the working environment. A library named "mobilerobot" embedded in Player was produced to control the designed robot. With this mobile robot library, the designed robot is controlled using client/server configuration where Player is a server providing interface between the user program and the devices on the designed robot such as sensor, motion controllers…. and the user program is a client that accesses and sends data to the designed robot through the Player server. The result of the obstacle avoidance algorithm based on potential field methods and the wall following one showed that the designed robot satisfied the objectives of a robot for educational purposes…

Keywords: *Mobile robot, Player/Stage, kinematic model, ultrasonic sensor, firmware, Potential field method, wall following, TCP/IP*

INTRODUCTION

During the last two decades, wheeled mobile robots have been increasingly presented in industrial and service robotics. For this reason, many universities have taught robotics in their master and PhD programs (Rawat & Massiha, 2004; Verner, Waks & Kolberg, 1999). Many robotic labs used mobile robots for educational purposes such as Amigo and Pioneer II from Active Media. These robots are equipped with sensors such as a ring of sonar, laser scan, camera to acquire the information of the environment. They can be controlled from a user program using C/C++ or Python on Window or Linux operating systems with client/server mechanism. These robots allow users to experiment with high level motion plan algorithms such as navigation and obstacle avoidance, localization and mapping… To respond to such need and enhance using the open source code to control robot, we have designed and built a low cost mobile robot that can be controlled on Player software. The designed robot can be control by a C++ program that uses *"mobilerobot"* library.

ROBOT ARCHITECTURE

The mobile robot uses a drive mechanism known as differential drive (Yamamoto & Yun, 1993). It consists of 2 drive wheels mounted on a common axis, and each wheel can independently be driven either forward or backward. The speed and orientation for the robot in mathematic can be written as (Yamamoto & Yun, 1993):

$$\begin{aligned} \dot{x} &= v_p.cos\theta \\ \dot{y} &= v_p.sin\theta \qquad (1) \\ \dot{\theta} &= \omega_\theta \end{aligned}$$

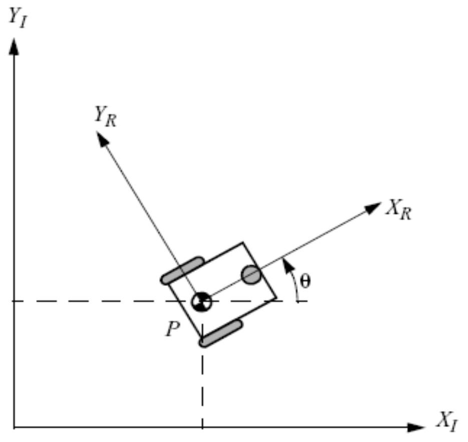

Figure 1: Differential Drive kinematics

The robot was equipped with a ring of 12 ultrasonic sensors SFR05 to provide the information about the obstacles and a digital compass, encoders to provide its orientation in the working environment. The layout in Figure 2 shows the distribution of the sonar. The ring of sonar sensor use alternating delay trigger technique to avoid interfere error. Linux laptop computer with a library named *"mobilerobot"* embedded in Player software was designed to control the designed robot. With this mobile robot library, the designed robot is controlled using client/server configuration where Player is a server providing interface between the user program and the devices on the designed robot such as sensor, motion controllers…. and the user program is a client that accesses and sends data to the designed robot through the Player server.

This designed robot has the diagram block of the robot controlled on the Player software as shown in Figure 3. Figure 4 shows the hardware block of the robot. It has AVR microcontrollers to handle data from peripheral components. The main AVR acquires the information from sonar sensors and saves this information on its memory in periods. Figure 5 shows the designed differential drive mobile robot.

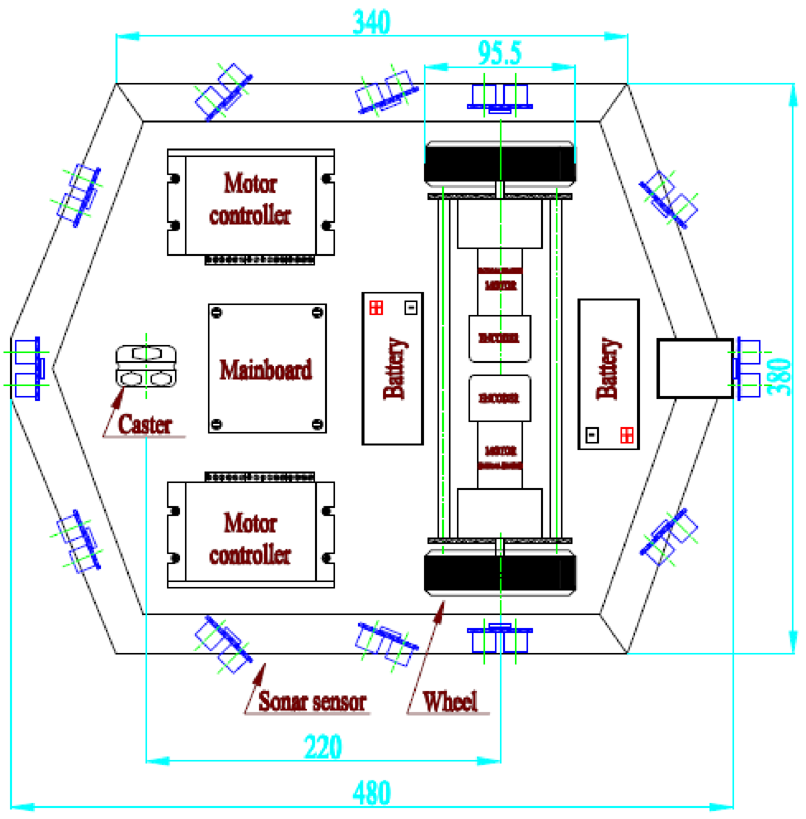

Figure 2. The layout of the robot frame and sonar

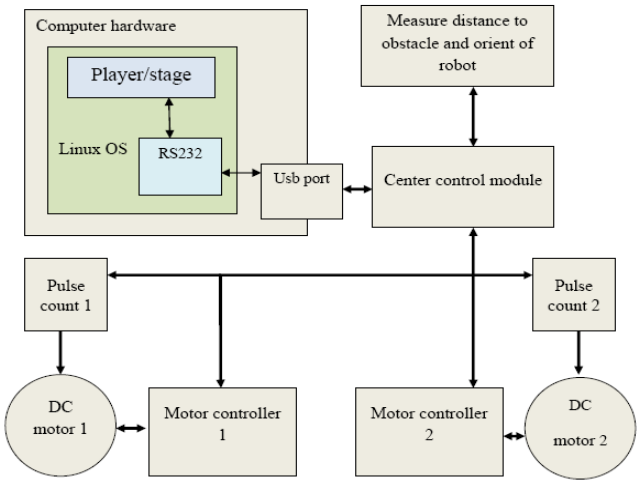

Figure 3. Diagram Block of mobile robot

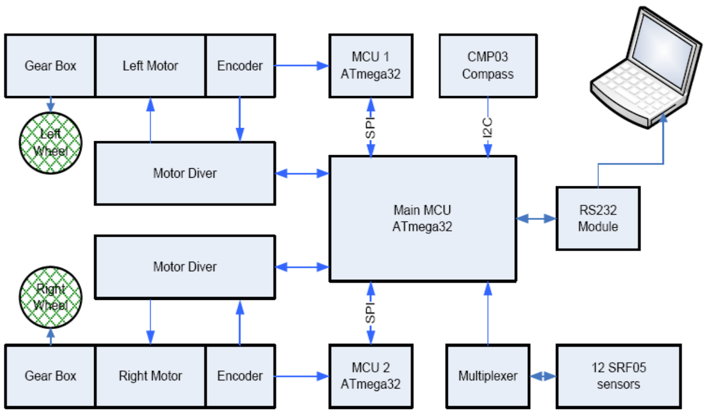

Figure 4. Structure hardware on the robot

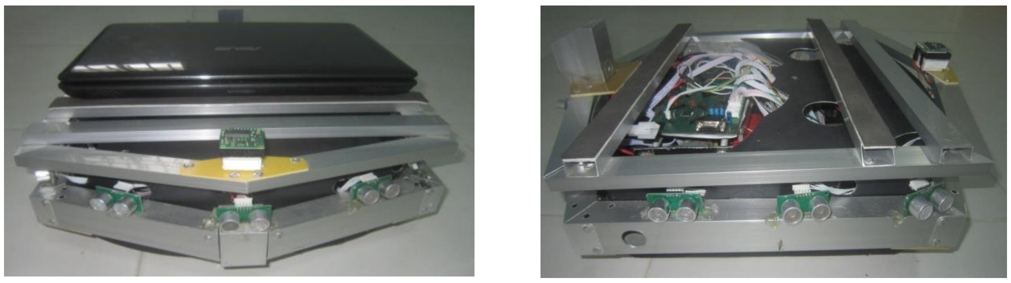

Figure 5. The Mobile robot designed

MOBILEROBOT LIBRARY ON THE PLAYER

The Player Project (Gerkey et al., 2003) creates Free Software that enables research in mobile robot and sensor systems. The Player robot server is probably the most widely used robot control interface in the world. It supports a wide variety of hardware, also C/ C++, and Python programming languages are officially supported. To control our robot, we built a plug-in mobile robot driver, named "*mobilerobot"* that inherits from Driver Class with some proxy such as *sonar* and *position2d* to get the information and set command velocity for the robot. A plug-in driver is compiled as a shared-object that is dynamically linked to the Player server's driver table at runtime.

The *mobilerobot.cfg* in Table 1 is an configuration file with some options, in which the shared-library for this plug-in driver is named *libmobilerobot_driver.so*. Once the driver has been instantiated and its central thread has been started by the *Setup()* function, the remaining processing is handled by the *Main()* method, which primarily consists of a loop that usually consists of 3 steps:

1. Sensor and state data are collected from serial class.
2. Collected data are published to the related devices
3. Incoming messages are processed.

Table 1. MobileRobot Class

```
# mobilerobot.cfg
driver
(
  name "mobilerobot"
  plugin "libmobilerobot_driver"
  provides ["position2d:0" "sonar:0"]
                       port "/dev/ttyUSB0"
)
```

Figure 6. Mobilerobot.cfg file with the plug-in driver

A brief list of the mobile robot class that needs to be implemented is given in Table 2.

Table 2. Mobilerobot Class

Name	**Brief Description**
Public Member Functions:	
MOBILEROBOTDriver(ConfigFile* cf, int section);	Constructor driver and file parsing.
int ProcessMessage(QueuePointer resp_queue, player_msghdr * hdr, void * data);	This Message handler function is called once for each message in the incoming queue.
Private Member Functions:	
void Main()	Threaded main loop,
int MainSetup()	Set up the device
void MainQuit()	Shut down the device.
*usbserial *serial*	USB serial Class
Server and Plugin-Specific Hooks	
Driver* MOBILEROBOTDriver_Init(ConfigFile* cf, int section)	Factory creation function that instantiates the Driver
void MOBILEROBOTDriver_Register(DriverTable* table)	Driver registration function that adds the driver into the given driver table
int player_driver_init(DriverTable* table)	Initiates the registration of the driver into the given driver table.

EXPERIMENTATION AND RESULT

As previously stated, this robot was designed and built to allow users to implement the high level control planner such as obstacle avoidance, localization and mapping, this section demonstrated two experiments that can be implemented on this platform, those are obstacle avoidance using potential field methods and wall following navigation.

Obstacle Avoidance based on Potential Method

Obstacle avoidance for mobile robot using potential field is simple but an efficient method. Potential field function was defined as below (Koren & Borenstein, 1991):

$$U(q) = U_{att}(q) + \sum_i U_{rep}(q) \qquad (2)$$

The artificial force $\vec{F}(q) = -\nabla U(q)$ at current configuration space, q denotes current position of the robot.

$$F_{total}(q) = -\nabla U_{rep} - \nabla U_{att} = F_{rep} + F_{att} \qquad (3)$$

The artificial attractive force from the goal:

$$F_{att}(q) = \begin{cases} -k_{att}(q - q_{goal}) & if \ \|(q - q_{goal})\| \le d \\ -dk_{att}\frac{(q-q_{goal})}{\|(q-q_{goal})\|} & if \ \|(q - q_{goal})\| > d \end{cases} \qquad (4)$$

The artificial repulsion forces of the obstacles:

$$F_{rep}(q) = \begin{cases} k_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q)}\nabla\rho(q) & if \ \rho(q) \le \rho_0 \\ 0 & if \ \rho(q) > \rho_0 \end{cases} \qquad (5)$$

Where k_{att}, k_{rep} is a constant gain, d is a constant that effect to the result of the attractive force, ρ_0 is a distance influent constant from the obstacle.

In potential field motion control as Figure 6, it is important to keep track of their $\vec{F}_{total}(q)$ forces, and their direction. The rule to produce velocity command for directing the robot can be written as below:

$$\omega_t = k(\alpha - \theta) \qquad (6)$$

where α is angle of $F_{total}(q)$, θ is the orient of the robot, k is a scale constant.

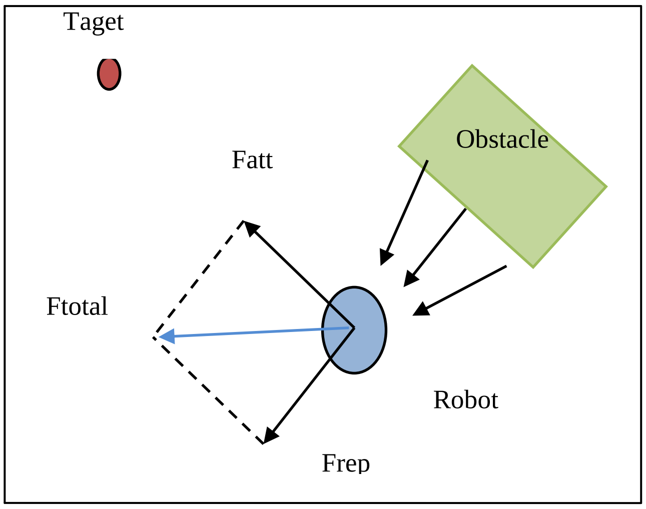

Figure 6. Potential field for target and one obstacle.

In this experiment, the parameters were chosen as follow:

$$\rho_0 = 60\ cm, k_{rep} = 95000, d = 30\ cm, k_{att} = 0.0066, k = 0.7, v_t = 0.1m/s$$

Figure 7 shows the working environment and the trajectory of the robot which was based on the information from wheel encoders, the position result was calculated by dead reckoning method and saved on computer memory. Figure shows the snapshots of the robot motion from this experiment.

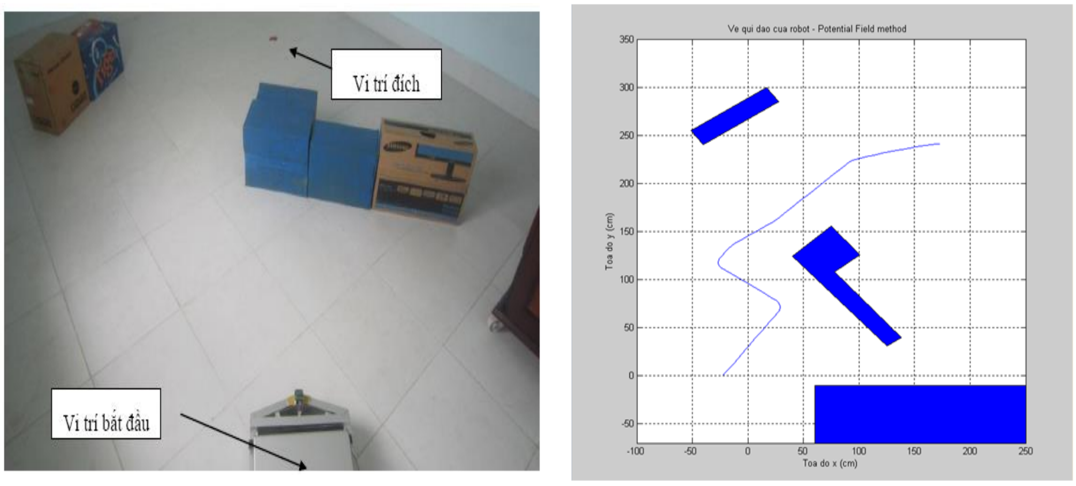

Figure 7. Experiment environment and the designed trajectory

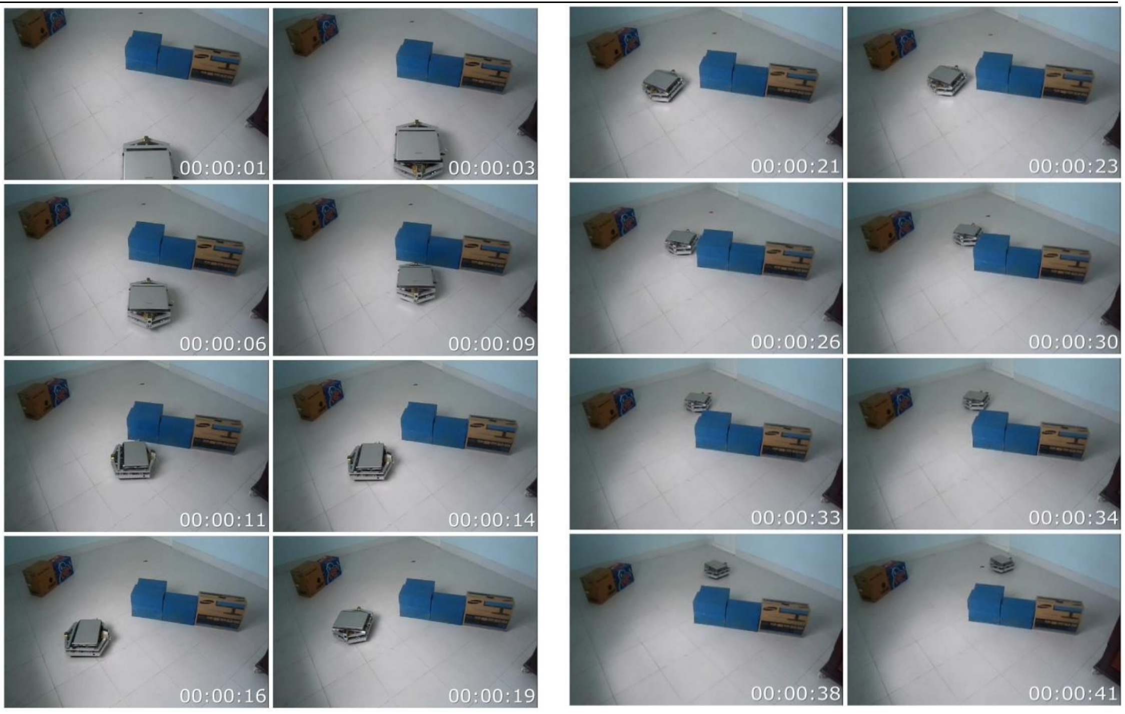

Figure 8. Snapshots of robot motion

Wall Following Navigation

When navigating in a working environment, the mobile robot may encounter deadlock caused by local minima. In this case, the wall following navigation method is an efficient strategy to help the robot fulfil its tasks. One sonar sensor on the ring of sonar is used to measure distance y from robot to the wall as shown in Figure 9.

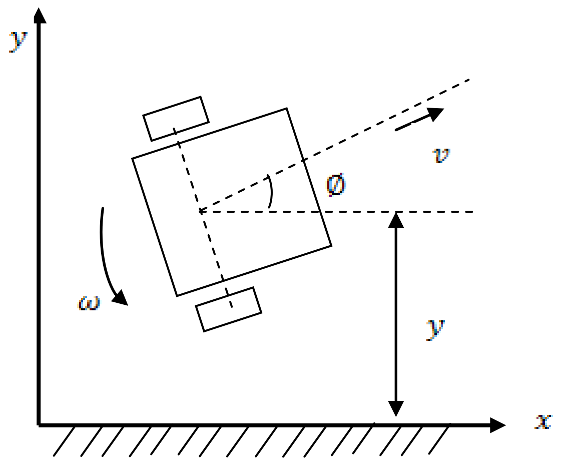

Figure 9. Wall following method

.

When an error occurs with $e = y - y_{set}$, the motion controller needs to keep $e \approx 0$. Error e was chosen as in a damped spring (Turennout et al., 1992).

$$F = ma = m\ddot{x} = -kx - k_f\dot{x} \qquad (7)$$

$$\ddot{e} + k_\theta \dot{e} + k_e e = 0 \qquad (8)$$

With θ small:

$$\dot{e} = v\sin\theta \approx v\theta$$

$$\ddot{e} = v\cos\theta\ \dot{\theta} \approx v\omega \qquad (9)$$

if $v = const$, ω was computed as below:

$$\omega = -k_\theta \theta - \frac{k_e}{v} e \qquad (10)$$

Where:

$$k_e = \frac{k_\theta^2}{4}$$

$$k_\theta = \sqrt{4k_e} \qquad (11)$$

k_e was chosen by experiment.

Figure 10 shows the robot working environment and its velocities when navigating around the room. Figure 11 shows the snapshot of this experiment.

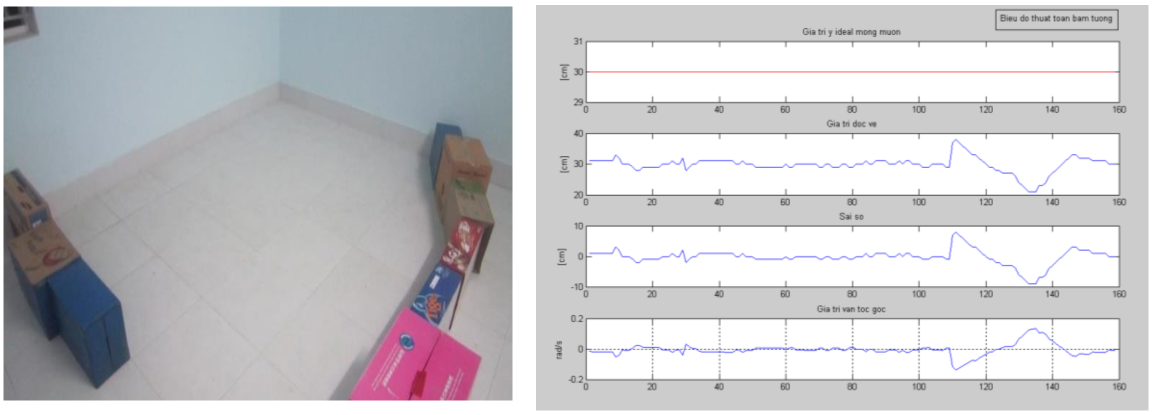

Figure 10. Result of the experiment

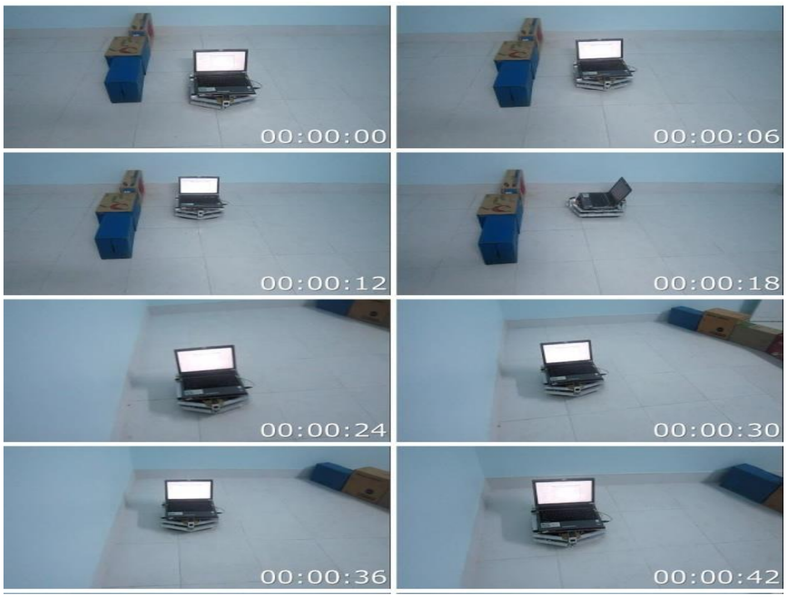

Figure 11. Snapshots of the experiment

CONCLUSION

This paper has shown the design and implementation of a mobile robot for educational purposes. The robot can be used to implement high level control algorithm in robotic research such as obstacle avoidance. The designed robot can be controlled in the same way as commercial mobile robots for educational purposes. The robot can be added with laser scanner and camera to perform tasks such as localization and mapping.

REFERENCES

Gerkey, B., Vaughan, R. and Howard, A. (2003). The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. *Proceedings of the International Conference on Advanced Robotics, 317-323*.

Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. *In Proc. IEEE Conf. Robotics and Automation, 1398–1404*.

Rawat, K.S. and Massiha, G.H. (2004), A hands-on laboratory based approach to undergraduate robotics education. *Proceeding of IEEE International Conference on Robotics and Automation,* Vol. 2, 1370 - 1374.

Verner, S., Waks and Kolberg, E. (1999). Educational robotics: An insight into systems engineering. *European Journal of Engineering Education*, 194-201.

Turennout, P., Honderd, G. and Schelven, L.J. (1992). Wall-following control of a mobile robot. *In IEEE International Conference on Robotics and Automation*, 280–285.

Yamamoto, Y. and Yun, X. (1993). Coordinating locomotion and manipulation of a mobile maniplator. *Recent trends in mobile robots*, Y.F. Zheng, Editor,World Scientfic: Singapor ; London, 157-181.