

*Date:* 2009-12-07  
*Student Name:* Jun Ling  
*Robot Name:* Follower  
*TAs:* Mike Pridgen  
Thomas Vermeer  
*Instructors:* Dr. A. Antonio Arroyo  
Dr. Eric M. Schwartz

# **Project “Follower”**

**Jun Ling**

**University of Florida  
Department of Electrical and Computer Engineering  
EEL-5666  
Intelligent Machines Design Laboratory  
Final Report**

## Table of Contents

1	Abstract.....	2
2	Introduction .....	2
3	Integrated Circuit.....	3
4	Mobile Platform.....	3
5	Actuation .....	5
6	Sensor .....	5
7	Feedbacks .....	5
8	Behaviors.....	6
9	Conclusion.....	8

### 1 Abstract

The aim is to develop a robot named *Follower*, whose primary behavior is to track a moving target. In this report, our emphasis is placed on how to develop a system to meet the robot requirements. The elaboration is functionally divided into two parts, namely, hardware platform integration and software development. Finally, we introduce the experimental layout and the demo strategy to test the behavior of *Follower*.

### 2 Introduction

The brain of *Follower* is an Atmel Xmega128 micro-processor chip and its eye is a Carnegie Mellon University (CMU) camera (henceforth CMUcam for short). *Follower* is equipped with three wheels: a caster omni-directional wheel at the front, and a pair of motor-driven wheels at the rear. Aside from CMUcam, other sensors used by *Follower* include two IR sensors for obstacle detection, a pair of line sensors to track black lanes. A LCD pad and a buzzer are used as debugging tools to monitor the system status during its operation. All the electronic components have been mounted on a plastic platform in a neat and compact manner.

Regarding the robot behavior, there are two modes *Follower* can engage. Briefly speaking, in mode 1, *Follower* tracks a moving target in open space. When mode 2 is engaged, *Follower* also tracks an object in a 2-lane situation; when a hostile vehicle moves toward *Follower*, it will dodge to avoid collision, changing lanes if necessary.

### 3 Integrated Circuit

The PCB, referred to as PVR board, is designed and developed by Mike Pridgen and Tom Vermeer, our TAs (hence the name PVR, short for Pridgen Vermeer Robotics). An Atmel Xmega128 micro-processor chip, which, as previously mentioned, serves as the brain of *Follower*, is mounted on the PVR board. Xmega128 is a powerful and general-purpose solution for robot development, and it is no doubt a viable candidate for meeting almost any peculiar requirements of a robot that can be reasonably developed within a single semester.

The main features of an Xmega128 micro-processor include:

- 12 PWM signals (5V supply)
- 8 ADCs (3.3V supply)
- 2 RS-232 terminals
- 1 SPI interface
- 4 8-bit digital IO ports
- 32MHz internal crystal

To meet the requirements of *Follower*, only a small subset of the abundant resources listed is used. For example, only 2 out of 12 PWM signals are used to control the motor speed, and 2 out of 8 ADCs are connected to IR sensors. 1 out of 2 RS-232 modules is allocated to interface with CMUcam.

The hardware architecture of *Follower* is shown in Figure 1. A module can be recognized as a system input if it is connected to Xmega128 with an arrow pointing to Xmega128, and as a system output with an arrow leaving Xmega128. Therefore, one observes that all the sensors (i.e., IR sensors, CMUcam and line sensors) are the system inputs. Outputs are the motor drivers, the LCD pad and the buzzer.

Recalling the brief description on the robot behavior presented in the previous section, we can see that the line sensors and the side IR sensor are not required in mode 1 (since there are no lanes or hostile vehicles in this mode); while mode 2 calls for teamwork of all the components involved.

### 4 Mobile Platform

Concerning that *Follower* has very few mechanical components (actually, the only mechanical component is motor), I did not use SOLIDWORK to design a wooden platform. Instead, the platform used is simply cut from a 4"×6" plastic photo frame that can be bought at Walmart. Figure 2 shows what finalized *Follower* looks like with the

major components being highlighted. We can see how well the photo frame fit into *Follower*.

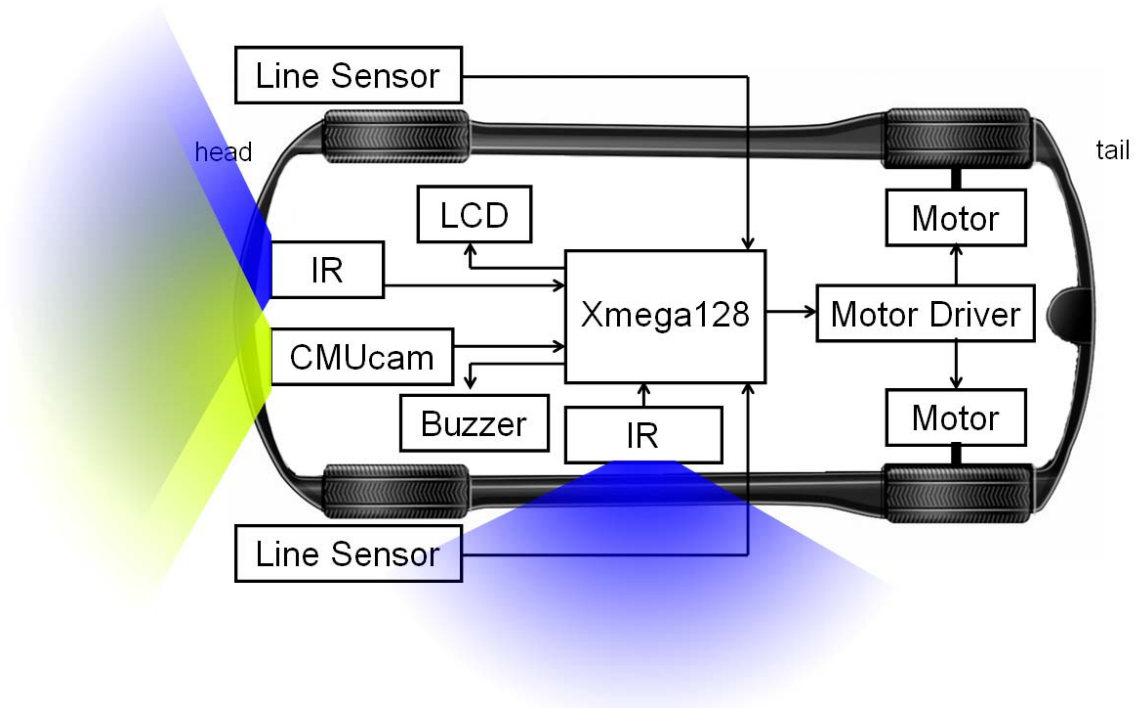


Figure 1: The hardware architecture of *Follower*.

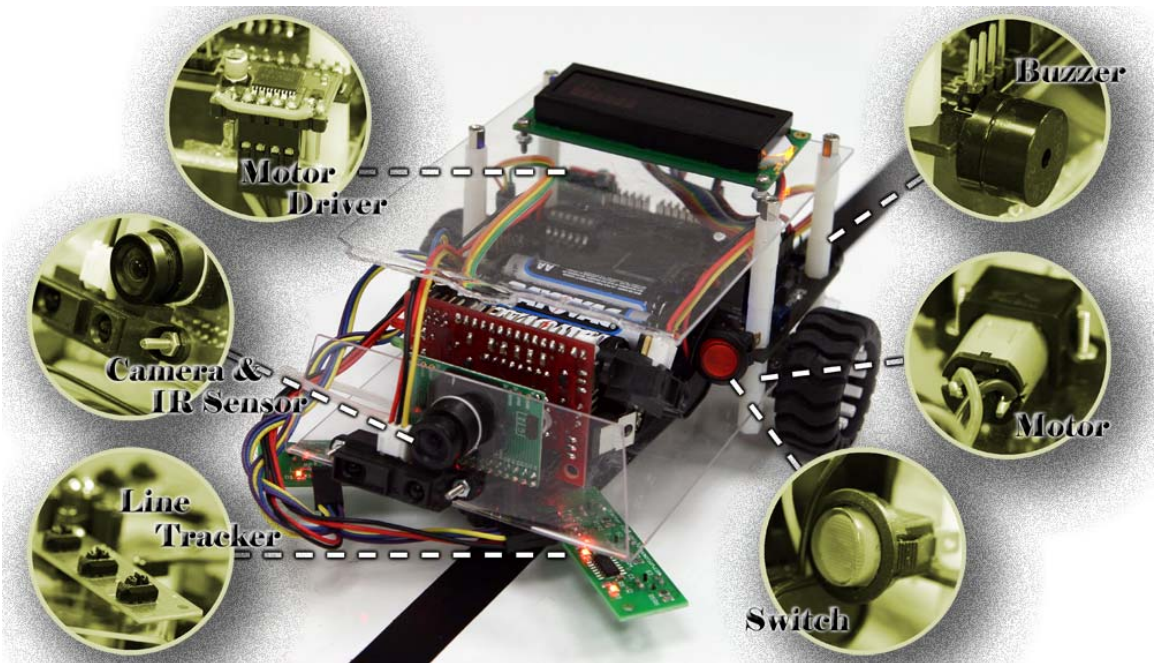


Figure 2: Structure of *Follower* with major components being labeled and highlighted.

## 5 Actuation

Two motors, in conjunction with the motor driver, are used to drive the rear wheels of *Follower*. To avoid messed wires, motor driver has been soldered directly to the PVR board on the pads initially connected to Port F, see Figure 2.

## 6 Sensor

As previously mentioned, all the sensors equipped on *Follower*, namely, the CMUcam, a pair of line sensors and two IR sensors, are the system inputs. A detailed treatment of CMUcam can be found in my “Special Sensor Report”. Therefore, we do not elaborate on CMUcam herein.

The model of the IR sensor is SHARP GP2Y0A21YK, and it is sold at \$9.95 at Sparkfun. *Follower* is equipped with two of them for obstacle detection: one in the front and the other on the left-hand side of *Follower*, see Figure 2. In both modes, the front IR will be used to measure the distance between *Follower* and the moving target. The side IR, however, is only used in mode 2 to detect a hostile vehicle moving toward *Follower*.

The line sensor is available online at Lynxmotion at a price of \$19.95. *Follower* is equipped with a pair of them to track black lanes in mode 2. Figure 2 shows that the line sensors are mounted to the underside of *Follower* toward the front.

## 7 Feedbacks

During the course of the robot development, in-circuit emulator (ICE), such as JTAG, is not accessible due to the budget problem. The absence of ICE renders the feedback modules, which can be used to monitor the status of the robot and facilitate debugging, crucial to the robot development. Fortunately, multiple feedback modules are readily available: some have already been soldered on the PVR board and some are peripherals that can be easily connected to the board.

On the PVR board, 2 LEDs are readily available. A blue one will be turned on whenever the board is powered, and the red one is programmable, whose function can be tailored to our requirements. *Follower* is equipped with two more feedback modules, namely, a LCD pad and a buzzer. The LCD pad displays a subset of ASCII characters on a  $16 \times 2$  screen. It is, of course, the most effective and productive feedback module to monitor the intermediate variables, check the sensor readings, etc. The PVR board has purposely laid out allocate a port to interface with LCD. Moreover, the availability of the

well-documented driver code makes this LCD peripheral almost plug-and-play. LCD is for visualize purpose and our ears are not left out: a buzzer is for audit function.

## 8 Behaviors

### 8.1 “self-checking and initialization” routine

After *Follower* has been powered on, it will first go through a “self-checking and initialization” routine. During this routine, Xmega128 tries to find all the peripheral sensors, and all the Xmega128 resources used by *Follower* are initialized by properly writing relevant control registers. Failure to pass this routine prevents *Follower* from proceeding with its normal behaviors (see below).

Only two situations can result in a failure: CMUcam cannot be found or the IR sensors cannot be found. Possible causes leading to a missing CMUcam would be wrong connection between CMUcam and the PVR board or an unpowered CMUcam system (note that the CMUcam system itself has a power switch, make sure it is turned on). For IR sensor, its readings will be polled multiple times, and system will declare a missing IR if all the readings are below a threshold. Consequently, in order to let the system detect IR sensor, we need to put something (our hand, for example) close enough to the front IR to allow at least one IR reading exceed the predefined threshold. To facilitate us to fix the problem (either missing CMUcam or missing IR sensor), a diagnosis will be displayed on LCD pad.

Note that, as previous mentioned, *Follower* can engage two different modes, and the mode selection task is also done during the “self-checking and initialization” routine via the readings from IR sensors. For example, if only the front IR has a reading exceeding the predefined threshold, it suggests that the mode 1 has been selected. On the other hand, if both the front and side IRs have readings exceeding the threshold, it is mode 2. See the demo video how it works.

The last thing before leaving the “self-checking and initialization” routine is to lock the target. This is done by grabbing the color feature of the target by launching a “TW” command to the CMUcam. Interested readers are referred to my “Special Sensor Report” for more details. After successfully passing the “self-checking and initialization” routine, *Follower* engage proper mode and behaviors accordingly.

### 8.2 Mode 1

Let’s recap *Follower*’s behavior in mode 1. *Follower* is supposed to track a red box in open space. Collision is not expected during the tracking process as *Follower* is designed

to keep a certain distance away from the target. Once again, note that there are no lanes or hostile vehicles involved in this mode, therefore the side IR sensor and the pair of line sensors are not activated. The software to implement this behavior is quite simple, and the pseudo code is given below:

```
while(1){  
    get distance information (from front IR);  
    get horizontal coordinate of the object (from CMUcam);  
    if (distance is OK)  
        track the object;  
    else  
        do something else;  
}
```

### 8.3 Mode 2

*Follower*'s behavior is more complex in mode 2. Beforehand, we should make two parallel black lanes on the ground. For the best performance, the lane-to-lane width is fixed at 5 inches approximately. The demo proceeds as follows: *Follower* is initially put on the left lane. After passing the "self-checking and initialization" routine, *Follower* tracks the red box on the left lane until a hostile vehicle approaches *Follower* from its left (we can simply move a hand toward it). *Follower* will detect this safety hazard via side IR sensor and then switch lanes to avoid collision. Then, on the right lane, *Follower* again follows the red box, and it attempts to switch back to the left lane when the safety hazard goes away.

The software that implements the above requirements is a finite state machine (FSM). Initially, *Follower* sticks to state 1 to track the box on the left lane. The detection of a hostile vehicle triggers the change in state from 1 to 2, in which *Follower* tracks the box on the right lane. And later on, when the safety hazard goes away, the state is switched back to 1.

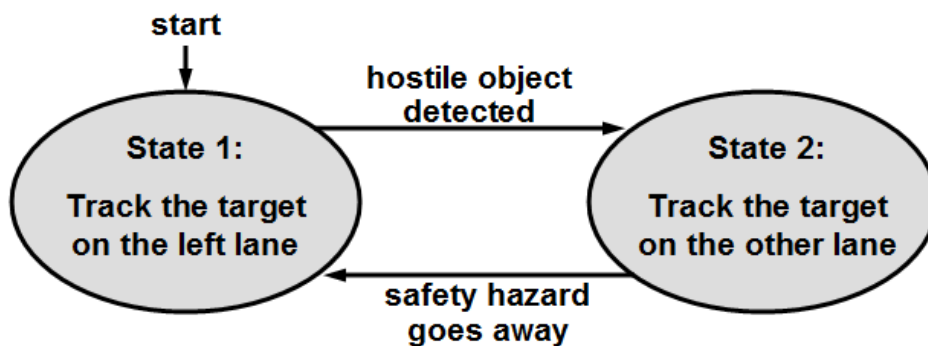


Figure 3: Finite state machine to implement *Follower*'s behavior in mode 2.

## 9 Conclusion

*Follower* is a one-semester project. Tight budget and rigid time schedule pose the major challenges. Therefore, it comes as no surprise that *Follower*, collectively with other robots developed by peer IMDLers, is fragile in nature and some sort of “dull” to the outside stimulus, be it by the unreliable system integration and the employment of cheap low-level sensors. Despite these defects inherent to the robot, it is really rewarding experience to witness the robot evolvement from scratch to its final version (strictly speaking, we did not start from scratch since, thanks to our TAs and instructors, PVR board is readily available, online materials are very helpful, and sample codes are well-documented and can be easily transplanted to our own code), to exchange ideas with peer IMDLers, and to establish my personal website to expose *Follower* to the outside world. Looking back, I still feel a little surprised at what I have achieved so far within one single semester.