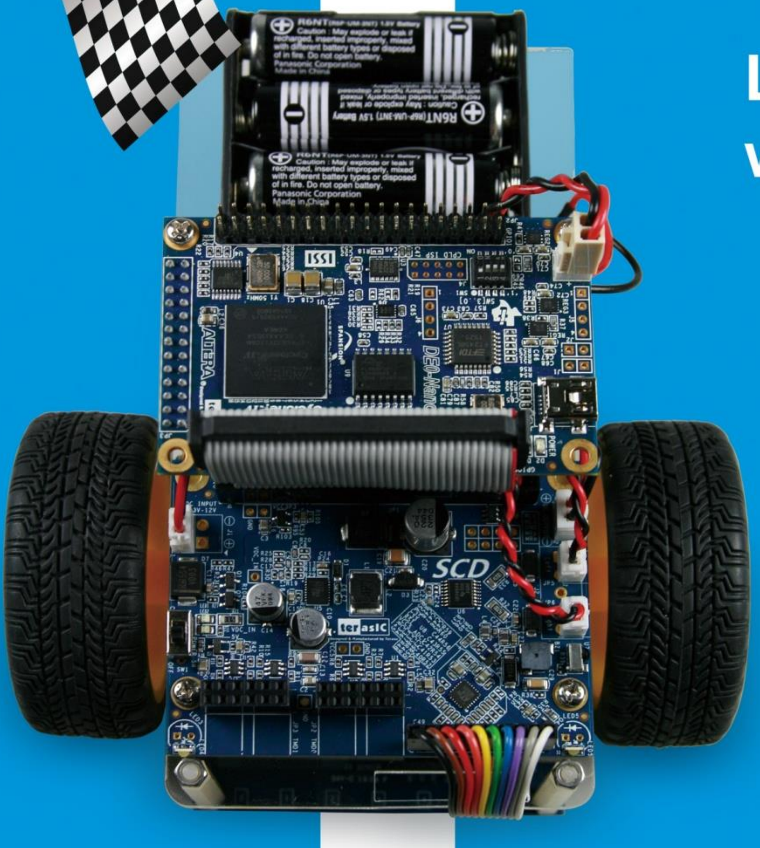
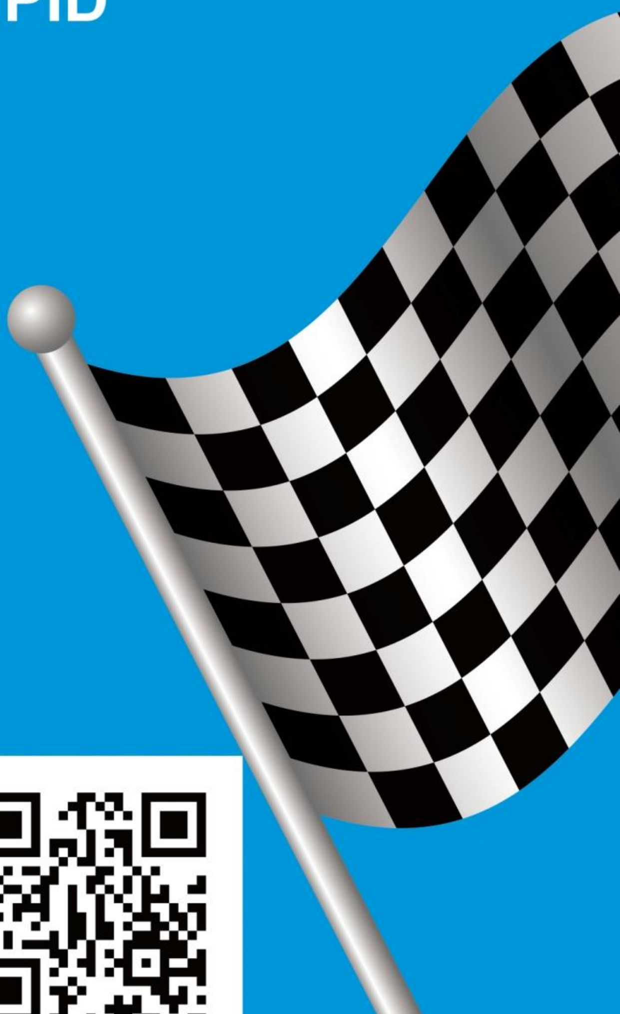
A-Cute Car

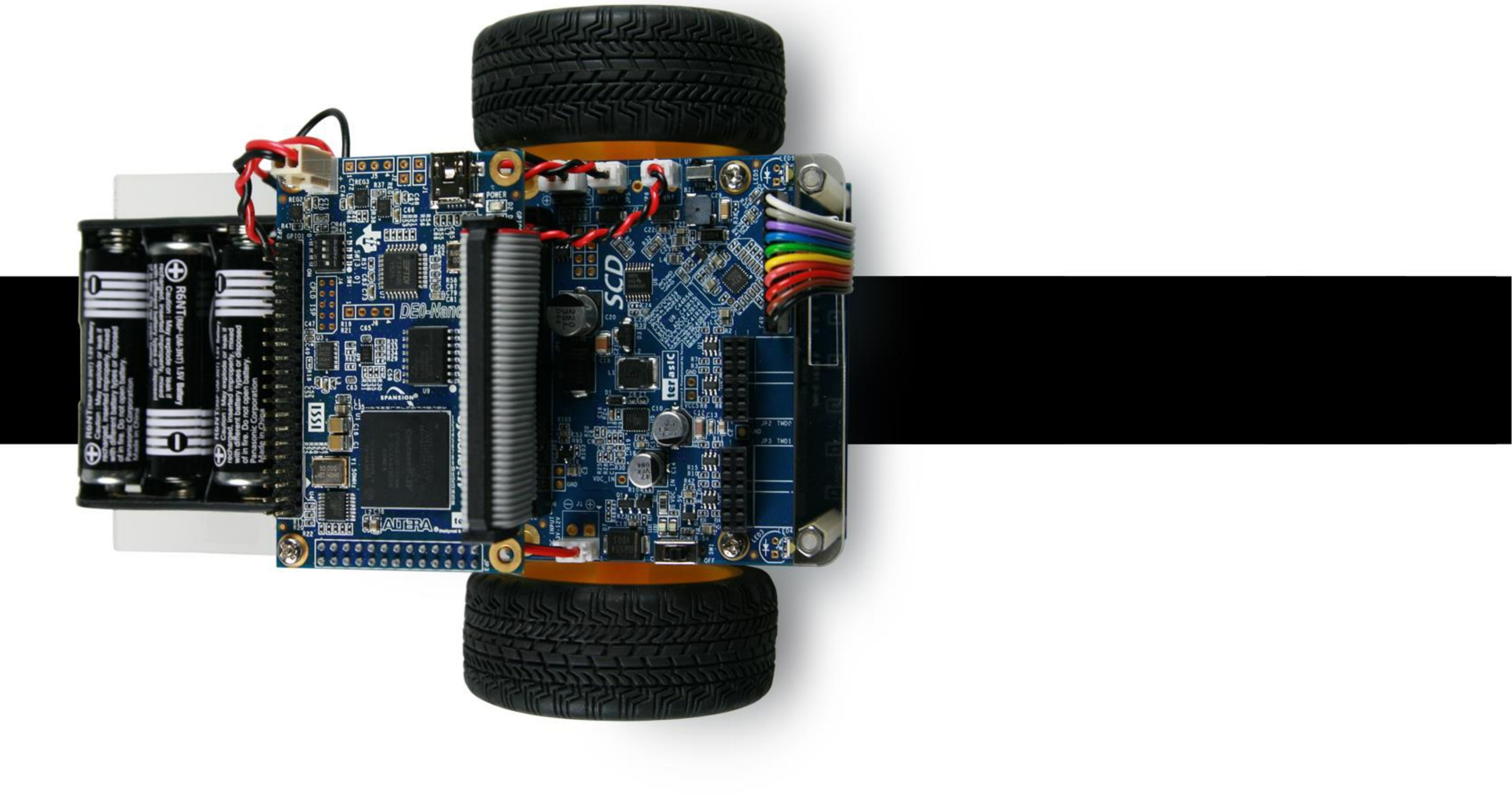
http://car.terasic.com



Line Following Robot with PID



terasIC
www.terasic.com



Abstraction

This document describes how to use the **PIDcontroller** to implement the LineFollowingfunction on the Terasic A-Cute Car. Besides the line following function, this demonstration also support IR remote control.

PID is abbreviation of proportional-integral-derivative. PID Controller uses a control loop feedback mechanism commonly used in industrial control systems.

Content

Abstraction 1

Content 2

Reference Design Overview 3

LTC2308 SPI Controller5

PWM Controller 6

IR Receiver Controller 7

PID Controller 8

Demo Setup 9

Rebuild Project 11

Improvement 11

Reference Design Overview

Figure 1 shows the Terasic A-Cute Car. The car is composed by three cards. DE0-Nano main card, SCD(Smart Car Daughter card) daughter card, and sensor daughter card. The SDC daughter card includes the lamp, buzzer, motor driver DRV8833, IR receiver, ADC chip LT2308, and TMD (Terasic Mini Digital) expansion header. The sensor daughter card includes seven Photo Interrupters used to track dark line(s) on a white background.

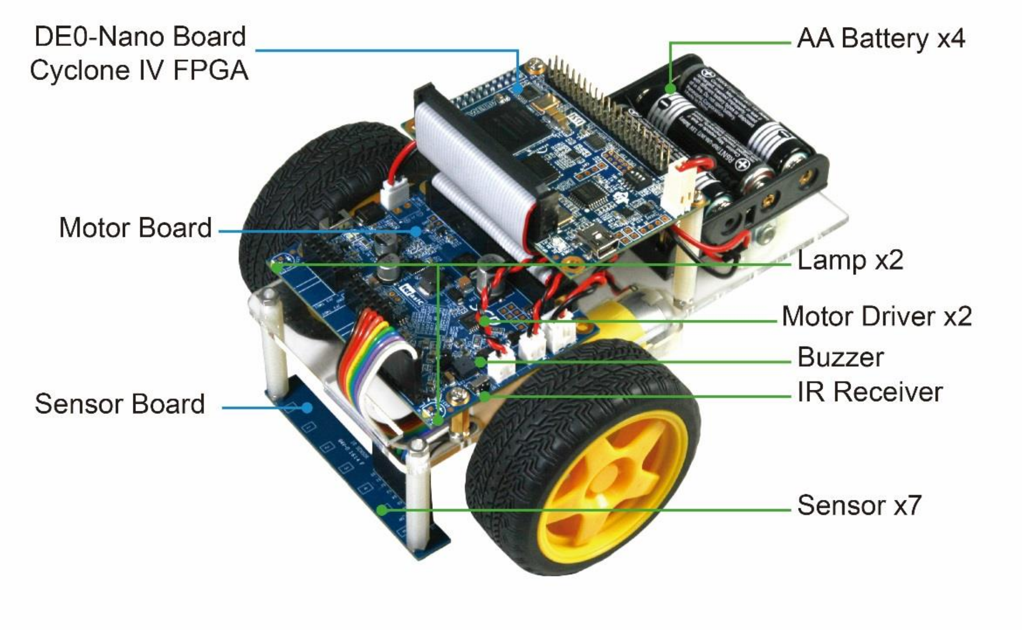


Figure 1 A-Cute Car

The hardware block diagram is shown in **Figure 2**. The PID controller is implemented in C++ code running on the Altera NIOS II Processor. The program is stored on the FPGA on-chip memory. The LTC2308 IP is used to read eight digitized value from the LTC2308 ADC chip through high speed SPI bus. The eight digitized values include one digitized value for the input power voltage and seven sensor values from the sensor board which containsseven Photo Interrupters used to track dark line(s) on a white background. The PWM IP is used to control the rotation speed and direction of DC motor. Each motor is controlled by a PWM controller. The 1K waveform IP is used to generate 1M frequency to drive the buzzer and the associated GPIO is used to control the beep sounds on or off switch.Left and right lamps are directly controlled by GPIO IP.The IR recevier is used to decode the recevied IR signal which is tramsmiited from the Terasicremoted controller.

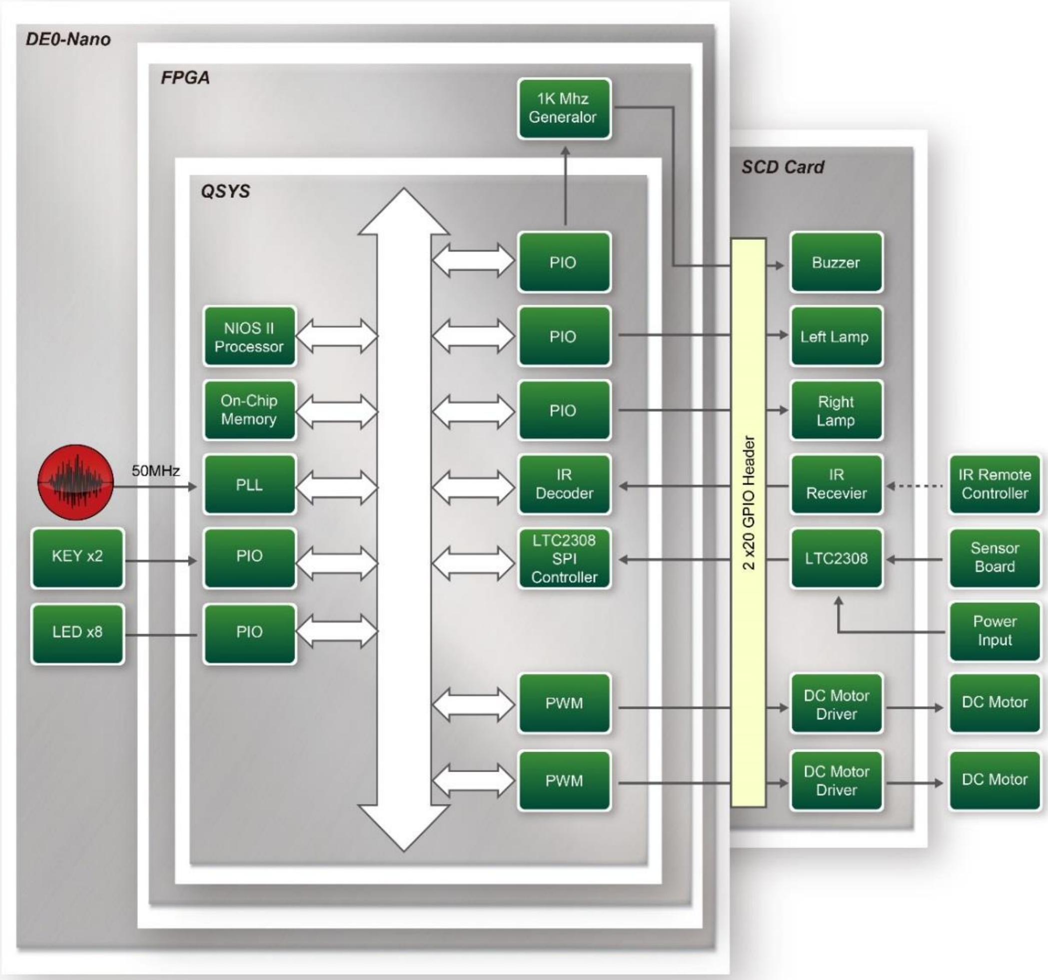


Figure 2 Hardware Block Diagram

Figure 2. Shows the software block diagram of the Line Following with PDI demonstration. The top block is C++ Structure Diagram whick interfaces with the QSYS IP by Altera Avalon Memory-Mapped (AVMM). In this demonstration, IORD and IOWR are used to communicate with the QSYS IP.

Main.cpp includes the line following PID control and simple PIO control for LED, KEY, Lamp and Buzzer. **CIrTx** object is used to handle the IR input. The **CIrTx** class is derived the **CQueue** class. All of received IR codes are pushed in to the queue;whose size is 8 in this demonstration. The main program get the IR codes by pop data from the queue. **CSensor** object is used to read the digitized ADC values from the ADC chip LTC2308. Main program reads seven sensor values and one input power value from this object. The **CCar** object is used to control the movement of theA-Cute Car. This object includes two **CDcMotor** objects which are used to control the two DC motor on theA-Cute Car. The

CDcMotor objects control PWM IP to control the DC motor speed and rotation direction.

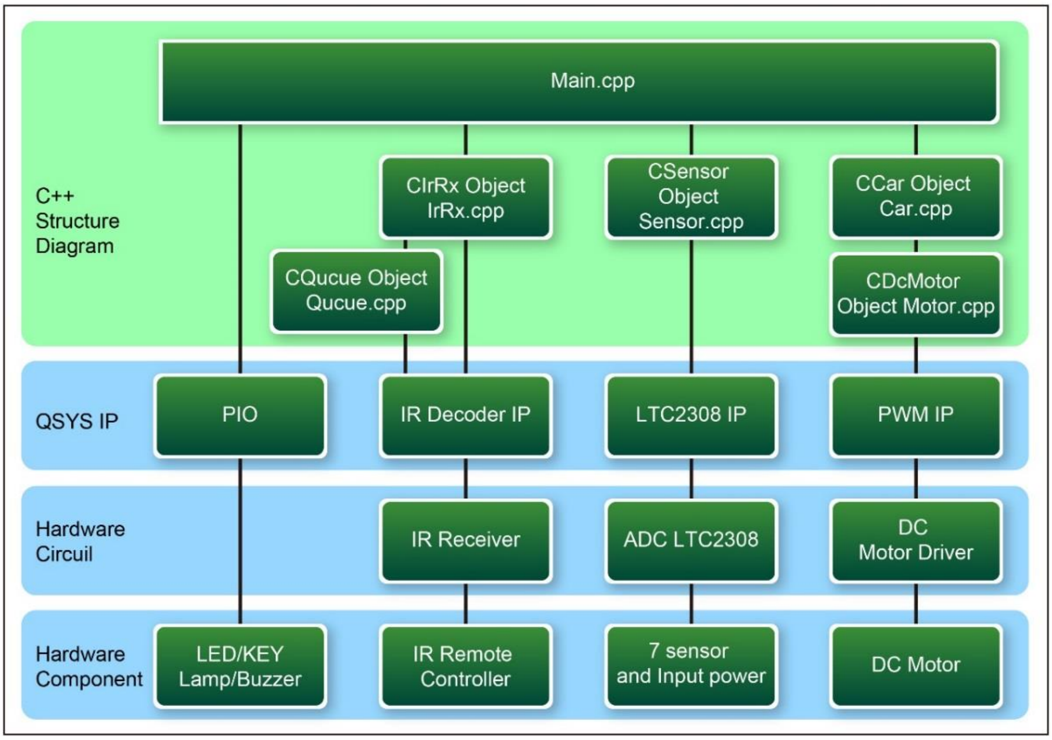


Figure 3 Software Block Diagram of Line Following with PDI

LTC2308SPI Controller

LTC2308 is a low noise, 500Ksps, 8-channel, 12-bit ADC chip. InTerasicA-Cute Car,the first seven channels are used to monitor the seven response values from the sensor board, and the last channel is used to monitor the input power voltage. The ADC is configured as single-ended, so the output value 0~4095 is represents voltage 0~4.095V, i.e., 1LSB represents 1mV.

The sensor value is low when sensor sees a white background, and value is high when sensor sees a black line. For last channel of the ADC chip can monitor larger range of input voltage, voltage division is applied such that only 1/4 voltage of the input power is connected to the ADC chip. The digitized value must be times 4.0 to get actually input voltage with unit mV.

In this demonstration, SPI bus is used between FPGA and LTC2308. LTC2308 SPI clock can be 40Mhz at maximal, however 20Mhz is used in this demonstration due to considering the GPIO cable reliability. The LTC2308

IPsource code located in the“ip\TERASIC_LTC2308” folder. The IP is enraptured as a QSYS Compliant IP. The register file of the IPis defined bellow.

Register Index	Register Name	Description	Read/Write
0	CS	Write: Bit 0 presents start bit, triggered by rising edge. Writing 0 then 1 to bit 0 start adc conversion. Read: Bit 0 presents read flag. Value 1 meansadc conversion is done and channel 0~7 data are ready on register 1~8.	RW
1	CH0	12 bit digitized value for channel 0	R
2	CH1	12 bit digitized value for channel 1	R
3	CH2	12 bit digitized value for channel 2	R
4	CH3	12 bit digitized value for channel 3	R
5	CH4	12 bit digitized value for channel 4	R
6	CH5	12 bit digitized value for channel 5	R
7	CH6	12 bit digitized value for channel 6	R
8	CH7	12 bit digitized value for channel 7	R

The CSensorC++ class defined in Sensor.cpp/h is designed to communicate with the LT2308 SPI hardware controller. The member functions ***ReadLineSenor*** and ***ReadInputPower*** can report the sensor response value and input power voltage individually.

PWM Controller

The PWM controller generate required duty cycle to control motor rotation speed. The IP source code is located in the folder “ip\TERASIC_DC_MOTOR_PWM”. The IP is enraptured as a QSYS Compliant IP. The register file of the IP is defined as bellows.

Register Index	Register Name	Description	Read/Write
0	TOTAL_DUR	32-bits integer. Represent the tick number of one PWM cycle.	RW
1	HIGH_DUR	32-bits integer. Represent the tick number of high level in one PWM cycle	RW
2	CS	Control Register	RW

		Bit0: Start bit. 1 : Start 0: Stop Bit1: direction bit. 1: forward, 0:backward	

The CDCMotorC++ class defined in Motor.cpp/h is designed to communicate with the PWM hardware controller. The member function ***SetSpeed*** with an input parameter ***fSpeed*** is designed to control motor speed and direction.***fSpeed*** value range is -100.0~100.0. Positive value presents forward rotary, and negative value presents backward rotary. 100 represent maximal speed for forward rotary, and -100 represents maximal speed for backward rotary. The ***SetSpeed*** function translate the input parameter ***fSpeed*** to required PMW parameters for the PWM controller. The translate formula also depends on the input voltage level which is used to drive the DC motor. The member function ***SetInputPower*** is designed for users to input the current input power voltage level. After setting motor speed, calling member function ***Start*** can start motor rotation. To stop motor rotation, developer can use the member function ***Stop***.

The CCar C++ class defined in Car.cpp/h is designed to control theA-Cute Car movement by controlling the two DC motors on theA-Cute Car. The member function ***SetSpeed*** is designed to setup car movement speed and direction. The member function ***Start*** is designed to start car moving, and the member function ***Stop*** is designed to stop car moving.

IR Receiver Controller

The IR Receiver IP receiving the input IR signal. When valid IR signal is received, the received IR scan code is stored in hardware FIFO and IRQ is asserted. The IP source code is located in the folder “ip\TERASIC_IRM”. The IP is enraptured as a QSYS Compliant IP. The register file of the IP is defined as bellows

Register Index	Register Name	Description	Read/Write
0	Scancode	Read: Read a received scan code from the FIFO. If FIFO is empty, e.g. no scan code is received, 0xdeadbeef is return. Write: Write any value to clear the interrupt flag.	RW

	When host interrupt handle routine handles the interrupt event, it should clear this interrupt flag.

The CIrRx C++ class defined in IrRx.cpp/h is designed to handle the received IR scan code. The CIrRx C++ class is derived from theCQueue C++ class defined in Queue.cpp/h. The received IR scancode will be stored in the Queue. The main program can use the member function ***IsEmpty*** to check whether any IR scancode is received. If there queue is not empty, main program can use the member function ***Pop*** to get the received scan code. To start receiving IR scan code, the main program should call the member function ***Enable*** to enable interrupt handling. To disable interrupt handing, main program can call the member function ***Disable***.

PID Controller

The PID Controller is implement in the Main.cpp.In this demonstration, only P and D are used. The PID code looks like the following. The **error** input will be used to generate new **output** value. The output value will be used to generate the **turn** value which is used to generate **LeftSpeed** and **RightSpeed** for the two motors on theA-Cute Car. In this demonstration, kp is 1.0 and kd is 8.0. (ki is 0.0)

```
intergral = intergral + error;
derivative = error - last_error;
last_error = error;
output = (kp * error + ki * intergral + kd * derivative); // PID
turn = output * 100.0;

LeftSpeed = Speed + turn;
RightSpeed = Speed - turn;

Car.SetSpeed(LeftSpeed, RightSpeed);
```

The above**error** value is calculatedby the following codes. The **szAdc**[] array present the seven values response from the seven sensors.

```
error = 0.0;
for(i=0;i<SENSOR_NUM;i++)
  error += szAdc[i] * (i+1);
```

```
fSum = 0;
for(i=0;i<SENSOR_NUM;i++)
fSum += szAdc[i];
if (fSum> 0){
  error /= fSum;
  error -= 4.0;  // mean is 4.0
}else{
  error = 0;
}
```

In this demonstration, the PID is looped 250 times per second. The loop count is determine by the ***nInterationInterval*** available. ***nInterationInterval*** is defined as in the following. To reduce the interval time (increase loop count per second), developer needs to speed up the c-code in the loop.

```
nInterationInterval = alt_ticks_per_second()/250;
```

Demo Setup

Here is the procedure to setup the demonstration:

- Set Power Switch on SDC card to OFF position (**Figure 4**)
- Insert four AA batteries
- Set Power Switch on SDC card to ON position
- This demonstration is the power on default code of theA-Cute Car. If the default code is erased, please execute test.bat in the folder for config the FPGA on de0-nano:

  A-Cute Car CD/DE0_NANO_LINE_FOLLOWER_PID/demo_batch

- Perform line following function:
  - Prepare your BLACK LINE map
  - Place theA-Cute Car on the BLACK LINE
  - Press KEY0 or KEY1 on DE0-Nano to start. Pressing 'Play' button onthe remote controller can also start the following function.
  - Removing the car from the track will stop the following function. Press "Power" button on the remote controller can also stop the following function.
- Perform IR remote control (**Figure 5**)
  - Place the car on the ground
  - Volume Up or Channel Up: car moves forward
  - Volume down or Channel Down: car moves backward
  - Adjust left: car turns left

- ✧ Adjust Right: card turns right
- ✧ A: car beep
- ✧ B: car two lamps active toggle
- ✧ C: car beeps and lamps active
- ✧ Number 0~9: adjusts car moving speed (0: mini, 9 maxi)
- ✧ Power: stop car
- ✧ Play: starts line following function

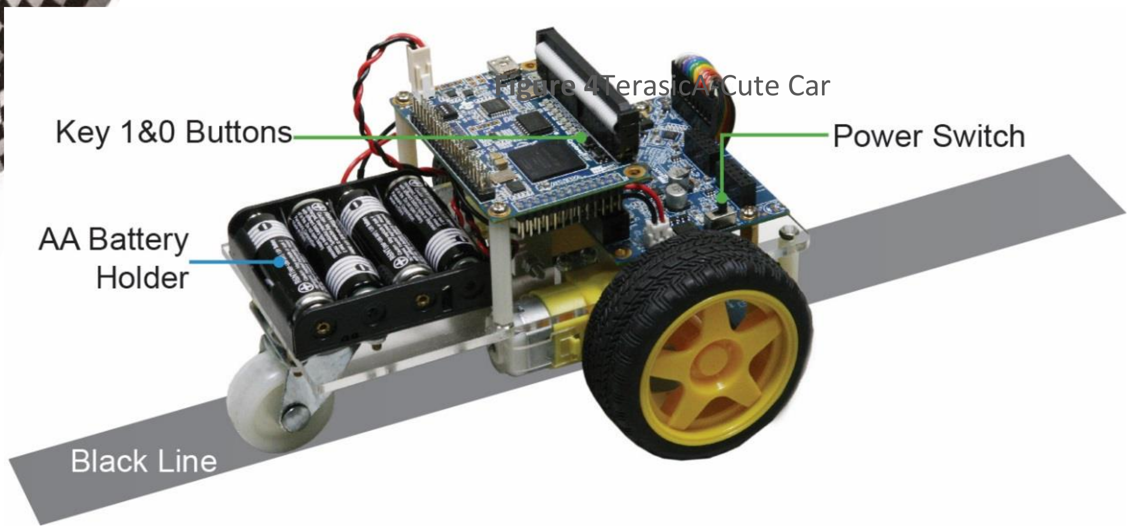


Figure 4 Terasic A-Cute Car

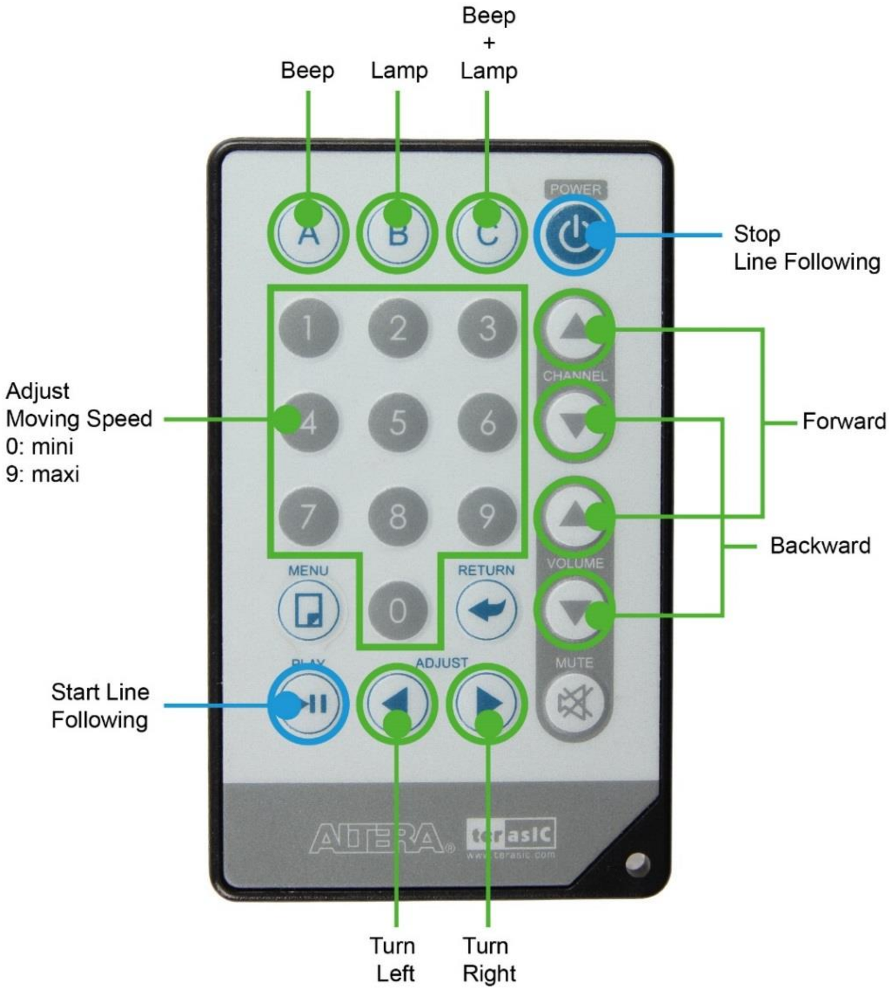
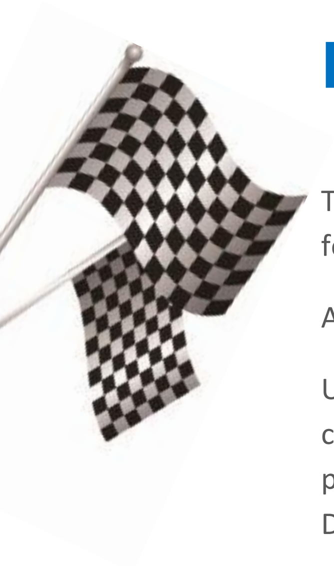


Figure 5 IR Remote Controller



Rebuild Project

The project is built by Quartus 15.0. The project source code is located in the folder:

A-Cut Car System CD/DE0_NANO_LINE_FOLLOWER_PID

Use Quartus to open the Quartusproject file DE0_NANO_SMART_CAR.qpf and click the menu item “Processing→Start Compilation” will start the compile process. When compilation is completed, an output file DE0_NANO_SMART_CAR.sof will be generated under the output_files folder.

The Nios II project is created by NIOS II 15.1 Software Build Tools for Eclipse. The project source code is located on the folder:

A-Cut Car System CD/DE0_NANO_LINE_FOLLOWER_PID/software

Launch NIOS II 15.1 Software Build Tools for Eclipse, the set above folder as workspace. In the Project Explore Window, right click “LINE_FOLLOWER_bsp[nios_system]” to popup a system menu, and select “NIOS II Generate BSP” to build the BSP. Then, right click “LINE_FOLLOWER” to popup a system menu, and select “Build Project” to generate binary file. When building is completed, an output file LINE_FOLLOWER.elf will be generated under the folder:

A-Cut Car CD/DE0_NANO_LINE_FOLLOWER_PID/software/LINE_FOLLOWER

Improvement

Here shows some methods that can improve the line following performance:

- Fine tune the kP and kD parameters in the PID controller.
- Implement the PID controller in RTL code to reduce the response time (increase loop count per second).
- Increase battery voltage to speed up motor.