CHAPTER 13

FPGA Robotics Projects

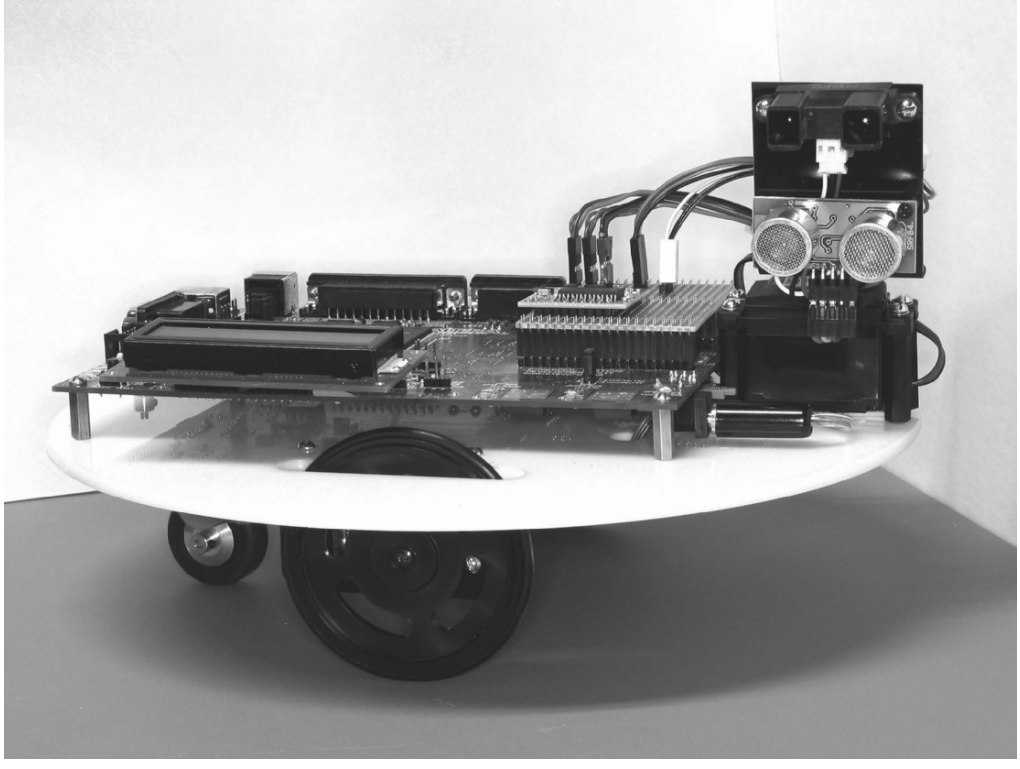


Photo: The FPGA-bot is a small robot controlled by an FPGA board

13 FPGA Robotics Projects

13.1 The FPGA-bot Design

The FPGA-bot shown in Figure 13.1 is a low-cost moving robotics platform designed for use the DE1, DE2, UP3, or UP2 board. The FPGA-bot is designed to be a small autonomous vehicle that is programmed to move in response to sensory input. A wide variety of sensors can be easily attached to the FPGA-bot.

The round platform is cut from plastic and a readily available 7.2V or 8.4V R/C rechargeable battery pack is used to supply power. Two diametrically opposed drive motors move the robot. A third inactive castor wheel or skid is used to provide stability. The robot can move forward, reverse, and rotate in place. Two relatively inexpensive radio control servos are used as drive motors. The FPGA is programmed to act as the controller. The R/C servos are modified to act as drive motors. The servos are controlled by timing pulses produced by the FPGA board.

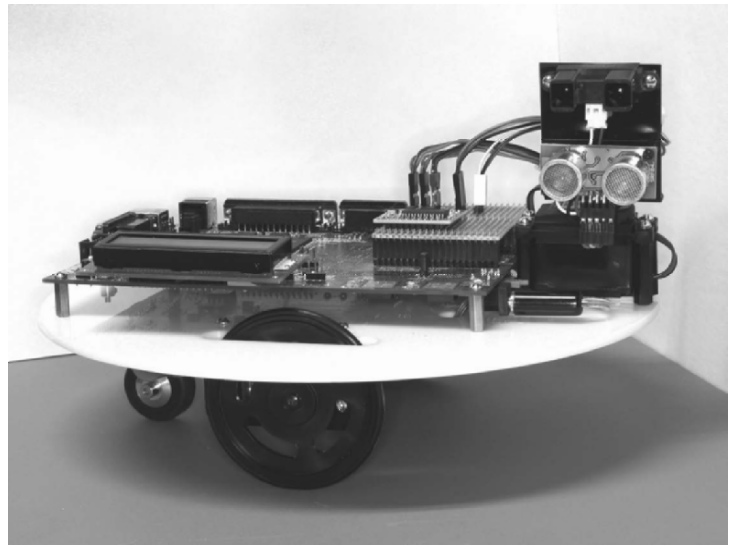


Figure 13.1 The FPGA-bot uses an R/C car battery and R/C servos for drive motors.

13.2 FPGA-bot Servo Drive Motors

A typical radio control servo is shown in Figure 13.2. Servos have a drive wheel that is controlled by a coded signal. The servo shown is a Futaba S3003 which is identical, internally, to the Tower TS53J servo. Radio control servomotors are mass-produced for the hobby market and are therefore relatively inexpensive and consistently available. They are ideally suited for robotics applications. Internally, the servo contains a DC drive motor (seen on

the left in Figure 13.2), built-in control circuitry, and a gear reduction system. They are small, produce a relatively large amount of torque for their size, and run at the appropriate speed for a robotics drive motor.

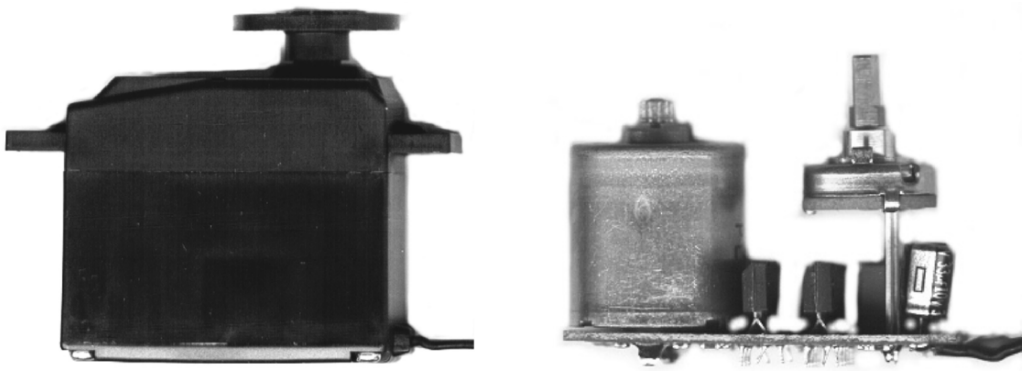


Figure 13.2 Left: Radio Control Servo Motor and Right: Servo with Case and Gears Removed.

The control circuitry of the servo uses a potentiometer (variable resistor) that is used to sense the angular position of the output shaft. The potentiometer is the tall component on the right in Figure 13.2. The output shaft of a servo normally travels 180-210 degrees. A single control bit is used to specify the angular position of the shaft. The timing of this bit specifies the angular position for the shaft. The potentiometer senses the angle, and if the shaft is not at the correct angle, the internal control circuit turns the motor in the correct direction until the desired angle is sensed.

The control signal bit specifies the desired angle. The desired angle is encoded using pulse width modulation (PWM). The width of the active high pulse varies from 1-2 ms. A 1ms pulse is 0 degrees, 1.5ms is 90 degrees and a 2 ms pulse is approximately 180 degrees. New timing pulses are sent to the servo every 20 ms.

13.3 Modifying the Servos to make Drive Motors

Normally, a servo has a mechanical stop that prevents it from traveling move than half a revolution. If this stop is removed along with other modifications to the potentiometer, a servo can be converted to a continuously rotating drive motor. Modifications to the servo are not reversible and they will void the warranty. Some robot kit vendors sell servos that are already modified.

To modify the servo, open the housing by removing the screws and carefully note the location of the gears, so that they can be reassembled later. The potentiometer can be replaced with two 2.2K ohm ¼ watt resistors or disconnected by cutting the potentiometer shaft shorter and setting it to the

center position so that it reports the 90-degree position. A more accurate setting can be achieved by sending the servo a 1.5ms pulse and adjusting the potentiometer until the motor stops moving. The potentiometer can then be glued in place with CA glue. In the center position the potentiometer will have the same resistance from each of the outside pins to the center pin. If the potentiometer is replaced with two resistors, a resistor is connected between each of the two outside pins and the center pin.

In some servos, there will be less mechanical play if the potentiometer is disabled by cutting the center pin and modified by drilling out the stop on the potentiometer so that it can rotate freely. The two resistors are then added to replace the potentiometer in the circuit.

The largest gear in the gear train that drives the output shaft normally has a tab molded on it that serves as the mechanical stop. After removing the screw on the output shaft and removing the large gear, the mechanical stop can be carefully trimmed off with a hobby saw, knife, or small rotary-grinding tool. The servo is then carefully re-assembled.

After modifications, if a pulse shorter than 1.5 ms is sent, the motor will continuously rotate in one direction. If a pulse longer than 1.5 ms is sent the motor will continuously rotate in the other direction. The 1.5 ms or 90-degree position is sometimes called the neutral position or dead zone. The drive signal to the motor is proportional, so the farther it is from the neutral position the faster it moves. This can be used to control the speed of the motor if the neutral position is carefully adjusted. A pulse width of 0 ms or no pulse will stop the servomotor.

A servo has three wires, +4 to +6 Volt DC power, ground, and the signal wire. The assignment of the three signals on the connector varies among different servo manufacturers. For Futaba servos, the red wire is +5, black is ground, and the white or yellow wire is the pulse width signal line. For JR and Hitec servos, the orange or yellow wire is the signal line and red is +5, and black or brown is ground.

On the FPGA-bot, the FPGA board must be programmed to provide the two timing signals to control the servo drive motors.

13.4 VHDL Servo Driver Code for the FPGA-bot

To drive the motors a servo signal must be sent every 20 ms with a 0, 1, or 2 ms pulse. The FPGA board is programmed to produce the timing signals that drive the motors. If no pulse is sent, the motor stops. If a 1 ms pulse is sent, the motor moves clockwise and if a 2 ms pulse is sent the motor moves in the reverse direction, counterclockwise. To move the FPGA-bot forward, one motor moves clockwise while the other motor moves counterclockwise. This is because of the way the motors are mounted to the FPGA-bot base.

In the code that follows, lmotor_dir and rmotor_dir specify the direction for the left and right motor. If both signals are ‘1’ the UP3 bot moves forward. The VHDL code actually moves one motor in the opposite direction to move forward. If both are ‘0’ the robot moves in reverse. If one is ‘1’ and the other is ‘0’, the UP3 bot turns by rotating in place. The two speed controls are

lmotor_speed and rmotor_speed. In the speed control signals, '0' is stop and '1' is run. A 1kHz clock is used for the counters in the module. The FPGAcore function, clk_div, can be used to provide this signal. Two more complex techniques for implementing variable speed control are discussed in problems at the end of the chapter. Acroname sells a low-cost optical encoder kit made by Nubotics that can be attached to standard R/C servo wheels and used for position feedback and more accurate motor speed control.

```
LIBRARY IEEE;
USE  IEEE.STD_LOGIC_1164.ALL;
USE  IEEE.STD_LOGIC_ARITH.ALL;
USE  IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY motor_control IS
     PORT      (clock_1kHz                              : IN      STD_LOGIC;
                lmotor_dir, rmotor_dir          : IN STD_LOGIC;
                lmotor_speed, rmotor_speed      : IN STD_LOGIC;
                lmotor, rmotor                  : OUT     STD_LOGIC);
END motor_control;

ARCHITECTURE a OF motor_control IS
     SIGNAL count_motor: STD_LOGIC_VECTOR( 4 DOWNTO 0 );
BEGIN
     PROCESS
          BEGIN
                              -- Count_motor is a 20ms  timer
               WAIT UNTIL clock_1kHz'EVENT AND clock_1kHz = '1';
                         IF count_motor /=19 THEN
                              count_motor <= count_motor + 1;
                         ELSE
                              count_motor <= "00000";
                         END IF;
                         IF count_motor >= 17 AND count_motor < 18 THEN
                                   -- Don't generate any pulse for speed = 0
                                   IF lmotor_speed = '0' THEN
                                        lmotor <= '0';
                                   ELSE
                                        lmotor <= '1';
                                   END IF;
                                   IF rmotor_speed = '0' THEN
                                        rmotor <= '0';
                                   ELSE
                                        rmotor <= '1';
                                   END IF;
                                   -- Generate a 1 or 2ms pulse for each motor
                                   -- depending on direction
                                   -- reverse directions between the two motors because
                                   -- of servo mounting on the FPGA-bot base
                         ELSIF count_motor >=18 AND count_motor <19 THEN
                                   IF lmotor_speed /= '0' THEN
                                        CASE lmotor_dir IS
                                             -- FORWARD
```

```
                                        WHEN '0' =>
                                                lmotor <= '1';
                                        -- REVERSE
                                        WHEN '1' =>
                                                lmotor <= '0';
                                        WHEN OTHERS => NULL;
                                        END CASE;
                                ELSE
                                        lmotor <= '0';
                                END IF;
                                IF rmotor_speed /= '0' THEN
                                    CASE rmotor_dir IS
                                        -- FORWARD
                                        WHEN '1' =>
                                                rmotor <= '1';
                                        -- REVERSE
                                        WHEN '0' =>
                                                rmotor <= '0';
                                        WHEN OTHERS => NULL;
                                        END CASE;
                                ELSE
                                        rmotor <= '0';
                                END IF;
                        ELSE
                                lmotor <= '0';
                                rmotor <= '0';
                        END IF;
    END PROCESS;
END a;
```

13.5 Low-cost Sensors for an FPGA Robot Project

A wide variety a sensors can be attached to the FPGA board. A few of the more interesting sensors are described here. These include infrared modules to avoid objects, track lines, and support communication between FPGA-bots. Other modules include sonar and IR to measure the distance to the nearest object and a digital compass to determine the orientation of the FPGA-bot. Most robots will need to combine or "fuse" data from several types of sensors to provide more reliable operation.

Signal conditioning circuits are required in many cases to convert the signals to digital logic levels for interfacing to the digital inputs and outputs on the FPGA board. Analog sensors will require an analog-to-digital converter IC to interface to the FPGA board, so these devices pose a more challenging problem. Small low-cost A/D ICs are available with SPI interfaces that require a minimal number of FPGA pins.

Sensor module kits are available and are the easiest to use since they come with a small printed circuit board to connect the parts. Sensors can also be built using component parts and assembled on a small protoboard attached to the FPGA-bot. Sensor modules are interfaced by connecting jumper wires to digital inputs and outputs on the FPGA board's J3 and J2 expansion header connector.

Sensors with a single output bit can utilize a simple control scheme, and for basic tasks the robot can be controlled using hardware as simple as a state machine. More advanced sensors that report actual distance, location, or heading measurements will likely require a processor core on the FPGA running a program that interprets sensor readings and implements the robot's control algorithm.

- **Line Tracker Sensor**

A line tracker module from Lynxmotion is shown in Figure 13.3. This device uses three pairs of red LEDs and infrared (IR) phototransistor sensors that indicate the presence or absence of a black line below each sensor. When the correct voltages are applied in a circuit, an IR phototransistor operates as a switch. When IR is present the switch turns on and when no IR is present the switch turns off. The LED transmits red light that contains enough IR to trigger the phototransistor.

Each LED and phototransistor in a pair are mounted so that the light from the LED bounces off the floor and back to the IR phototransistor. The LED and IR sensor must be mounted very close to the floor for reliable operation. Black tape or a black marker is used to draw a line on the floor. The black line does not reflect light so no IR signal is returned. Three pairs of LEDs and IR phototransistor sensors produce the three digital signals, left, center, and right. The FPGA-bot can be programmed to follow a line on the floor by using these three signals to steer the robot. The mail delivery robots used in large office buildings use a similar technique to follow lines or signal cables in the floor.

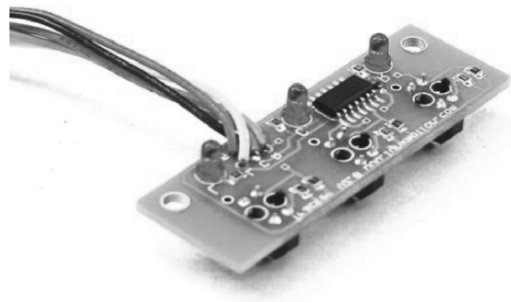


Figure 13.3 – Three LEDs and phototransistors are mounted on bottom of the Line Tracker board.

- **Infrared Proximity Detector**

An IR proximity sensor module from Lynxmotion is seen in Figure 13.4. The FPGA-bot can be outfitted with an infrared proximity detector that is activated by two off-angle infrared transmitting LEDs. The circuit utilizes a center-

placed infrared sensor (Sharp GP1U5) to detect the infrared LED return as seen in Figure 13.5. The Sharp GP1U5 was originally designed to be used as the IR receiver in TV and VCR remote control units. From the diagram, one can see that the sensitivity of the sensor is based on the angle of the LEDs. The LEDs can be outfitted with short heat-shrink tubes to better direct the infrared light forward. This prevents a significant number of false reflections coming from the floor. The IR sensor will still occasionally detect a few false returns and it will function more reliably with some hardware or software filtering.

Figure 13.4 IR Proximity Sensor Module – Two IR LEDs on sides and one IR sensor in middle.

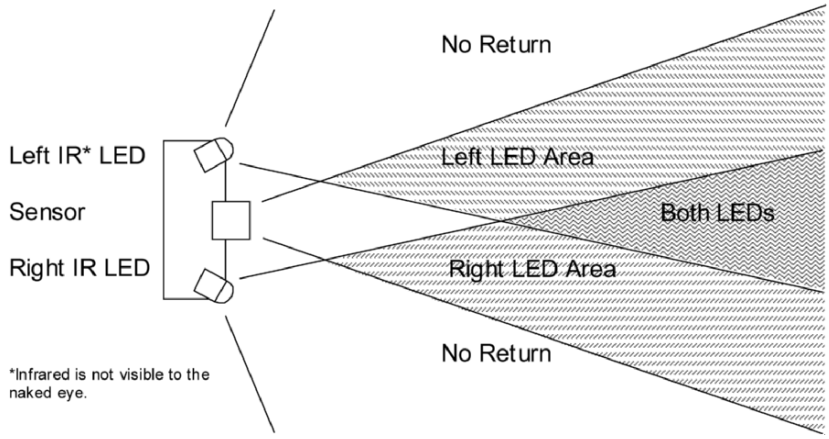


Figure 13.5 Proximity detector active sensor area.

As seen in Figure 13.6, the circuit on the IR proximity module utilizes a small feedback oscillator to set up a transmit frequency that can be easily detected by the detector module. This module utilizes a band-pass filter that essentially filters out ambient light. Some older first generation electronic ballasts used in commercial fluorescent lights can interfere with the IR sensors since they operate at the same frequency as the filter. Newer ballasts now operate at a higher frequency since they also caused problems with IR TV remote control signals.

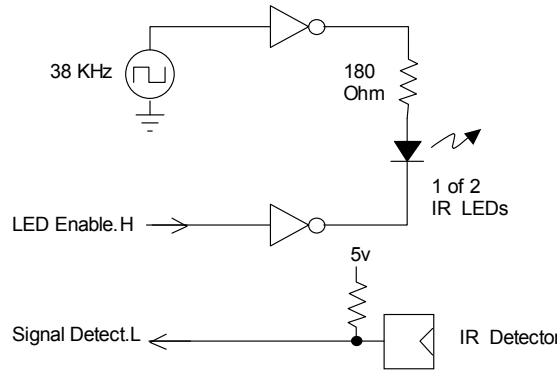


Figure 13.6 Circuit layout of one LED and the receiver module on the infrared detector.

In Figure 13.6, when the Left_LED Enable signal is High, the Low side of the IR LED is pulled to ground. This forces a voltage drop across the LED at the frequency of the 5v to ground oscillating signal. In other words, the LED produces IR light pulses at 38 kHz. Using a 38 kHz signal helps reduce noise from other ambient light sources.

Since the IR detector has an internal band-pass filter centered at 38 kHz, the detector is most sensitive to the transmitted oscillating light. The 5v pull-up resistor allows the IR Detector's open collector output to pull up the SOUT signal to High when no IR output is sensed. To detect right and left differences, the right and left LEDs are alternately switched so that the detected signals are not ambiguous. If both the left and right LEDs detect an object at the same time, the object is in front of the sensor.

If the IR sensor was built from component parts, a hardware timer implemented on the UP3 board could be used to supply the 38-40 kHz signal. Similar IR LEDs and IR detector modules are available from Radio Shack, #276-137B, and Digikey, #160-1060. Assuming two FPGA-bots are equipped with IR sensor modules, it is also possible to use this module as a serial communication link between the robots. One FPGA-bot transmits using its IR LED and the other FPGA-bot receives it using its IR sensor. To prevent interference, the IR LEDs are turned off on the FPGA-bot acting as a receiver. Just like an IR TV

remote, the IR LED and sensor must be facing each other. Bandwidth is limited by the 38kHz modulation on the IR signal and the filters inside the IR detector. (An IR sensor strip that converts IR to visible light is available from Radio Shack. This sensor can be used to confirm the operation of IR LEDs.)

- **Wheel Encoder**

  The Nubotics WW01 WheelWatcher incremental quadrature encoder system from Acroname is shown in Figure 13.7. This low-cost electronics board bolts onto the top of a standard-size R/C servo. The adhesive-backed codewheel attaches to a wheel mounted on the servo's output shaft. Two pairs of optical emitters and receivers bounce light beams off of the codewheel.

  Note that there are 32 black stripes on the reflective codewheel. When the wheel is rotating, the encoder produces two series of digital pulses that are 90 degrees out of phase. When one of the pulses changes twice before the other pulse changes, the direction has been reversed. 128 clock pulses per revolution are produced and a separate direction signal indicates the current direction of rotation. By counting pulses with a counter or by accurately measuring the time between individual pulses using a fast hardware counter on the FPGA, it is possible to more accurately control the position and velocity of the servo motor. When used on robot drive motors, this optical encoder feedback provides more accurate position and speed control for the robot.

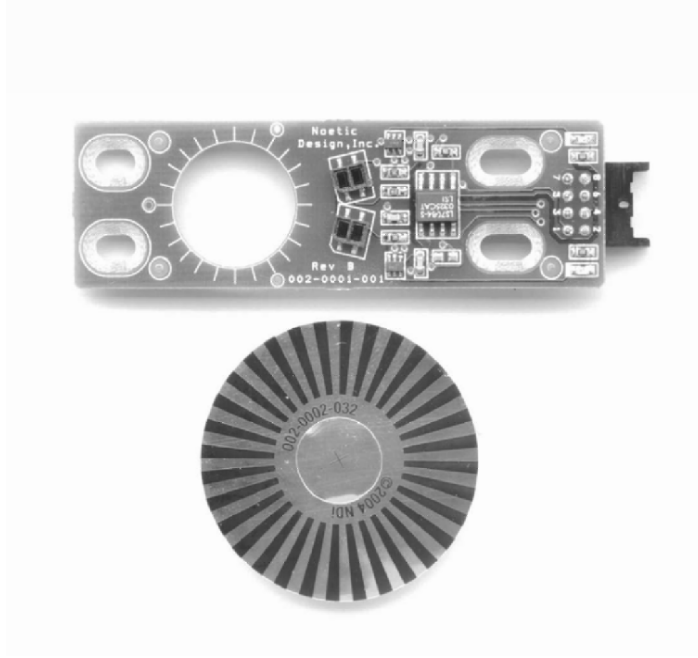


Figure 13.7 Nubotics WW-01 Wheel Watcher Incremental Encoder System.

- **Sonar Ranging Units**

  The Devantech SRF10 Sonar Module is shown in Figure 13.8. This device uses ultrasonic sound waves to measure distances from a few inches to around 35 feet. They are widely used in robotics. The timing of the sound echo indicates distance to the nearest object. The transducer first functions as a transmitter by emitting several cycles of a ultrasonic signal, and then functions as a receiver to detect sound waves returned by bouncing off nearby objects. Even though ultrasound is inaudible, the transducer also generates a slight audible click each time the device transmits. The beamwidth is rather wide, and several sonar modules facing in different directions are commonly used.

  The time it takes for the ultrasonic echo signal to return is measured using an IC mounted on the back side of the board. This time is converted to distance since sound travels out and back at 0.9 ms per foot. Only around 10-20 samples per second are possible with the device since it takes time to wait for echoes to return. Some sonar modules require external hardware to measure the pulse timing to produce the distance to target. The SRF10 device operates off +5V DC, and it sends distance measurements back to the host using an I^2C bus. A number of other similar Sonar modules are available.

Figure 13.8 Devantech SRF10 Ultrasonic Range Finder.

- **IR Distance Sensors**

  The Sharp GPD2D02 seen in Figure 13.9 is an IR device that can provide distance measurements similar to the slightly more expensive sonar sensor. This sensor has a shorter range of 10 to 80 cm (~ 4 to 32 inches). The distance is output by the sensor on a single pin as a digital 8-bit serial stream.

Figure 13.9 Sharp IR Ranging Module.

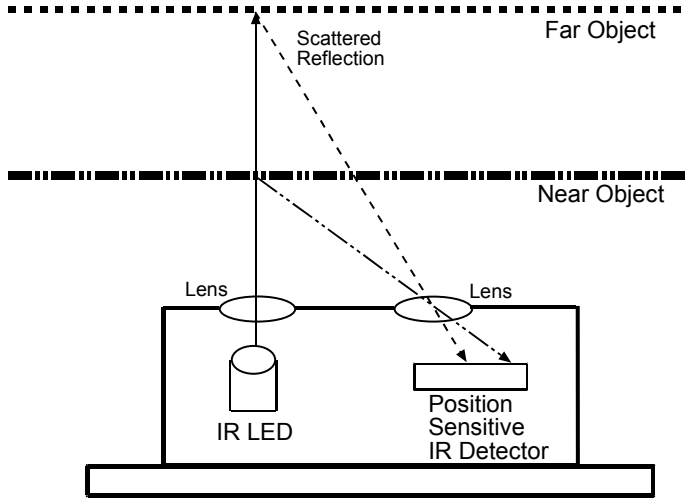


Figure 13.10 Operation of Sharp IR Ranging Module.

As shown in Figure 13.10, internally the GPD2D02 contains an IR LED and a position-sensitive IR detector. The IR LED transmits a modulated beam of infrared light. When the light strikes an object, most of the light will be reflected back to the LED. Since no surface is a perfect optical reflector, scattering of the IR beam occurs at the surface of the object and some of the light is reflected back to the position sensitive detector. By comparing the near and far object beams shown in Figure 13.10, it is apparent that the position at which the scattered reflected IR beam hits the detector is a function of the reflection angle.

The 8-bit integer value reported by the sensor in cm is approximately

$$1000 * \tan^{-1}\left(\frac{1.9}{DISTANCE}\right) + offset\,.$$

The constant 1.9 is the distance between the lenses in cm. The offset is the no-object present value returned by the sensor. This offset constant can vary by as much as 17 between different sensors and has a typical value of 25. Note that a close object reports a larger value and a distant object reports a smaller value. Objects closer than 10cm will report an incorrect value and should be avoided by placing the sensor away from the edge of the robot. Large objects beyond 80 cm can sometimes report an incorrect value that makes them appear closer.

A special connector (Japan Solderless Terminal #S4B-ZR) is required to connect to the GPD2D02. If desoldering equipment is available, the small connector can also be desoldered from the sensor and wires attached directly to the sensor.

In addition to +5V and ground pins, the sensor has an input, Vin, and a serial output, Vout. Vin is an input to the sensor that clocks out the serial data on Vout. When Vin is Low for around 70 ms, the sensor takes a reading. When a reading is available, Vout goes High.

On each of the next eight falling clock edges of Vin, the sensor will output a new data bit. The eight data bits should be clocked into the FPGA on the rising edges of Vin (when they are stable). When clocking out the data, the clock period on Vin should be 0.4 ms or less. The eight data bits are clocked out in high to low order. If Vin is not dropped Low within 1.5 ms after clocking out the final data bit, the sensor shuts down to save power. A shift register can be used to assemble the data bits. The demo program ir_dist.bdf on the DVD contains a VHDL-based IP core for use with the GP2D02 sensor.

The sensor's Vin pin is an open-drain input. Open-drain or open-collector inputs should never be driven High. An FPGA's tri-state output pin can be connected directly to an open-drain input, if the tri-state output is never driven High. When Vin should be High, tri-state the FPGA's output pin and when the output should be Low, drive the output pin Low with the tri-state gate turned on with a low output.

Open-drain or open-collector inputs contain an internal pull-up resistor to +5V. Multiple open-drain (open-collector) outputs can be tied together to a single open drain (open-collector) input to perform a wired-AND operation. Any one of the outputs can pull the input Low. If no output pulls the signal Low, a single pull-up resistor forces the input High. This wired-AND operation occurs just by tying the open-drain (open-collector) outputs together and no physical AND gate is needed. In negative logic, a wired-OR operation occurs.

Normal gate outputs cannot be connected. This wired-AND logic only works because these gates have special output circuits that do not contain a transistor that forces the input High. This transistor is present in normal gate outputs. If a normal gate output is connected to other open-drain (open-collector) outputs,

its transistor could turn on to force the input High at the same time another gate's output transistor turns on to force it Low. This would short the power supply to ground drawing excessive current that might damage the devices. An analog and longer range version of this IR distance sensor are also available.

- **Magnetic Compass Sensors**

  Various electronic components are available that detect the magnetic field of the earth to indicate direction. A low-cost digital compass sensor is shown in Figure 13.11. The Dinsmore model 1490, often used in electronic automobile compasses, is a combination of a miniature rotor jewel suspended with four Hall-effect (magnetic) switches. Four active-low outputs are provided for the four compass directions. When the module is facing North, the North output is Low and the other three outputs will be High. Eight directions are detected by the device, since two outputs can become active simultaneously. In this way, the device can indicate the four intermediate directions, NE, SE, SW, and NW. NE for example activates the active-low North and East outputs. The device can operate off +5V.

  Mount any compass device as far away from motors as possible to avoid magnetic interference from the magnets inside the motor. Four 2.2K ohm pull-up resistors to +5V are required to interface to the UP3 board, since the four digital output pins, N, S, E, and W, all have open-collector outputs. Just like a real compass, a time delay is needed after a quick rotation to allow the outputs to stabilize. If the compass module leads are carefully bent, the compass module and the four required pull-up resistors can be mounted on a standard 20-pin DIP, machined-pin, wire-wrap socket and connected to the UP3 header socket.

  An analog version of the device is available with 1-degree accuracy, but it requires an analog-to-digital conversion chip or signal phase timing for interfacing.

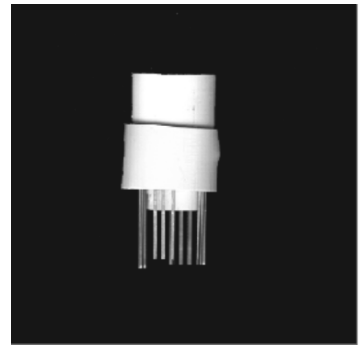


Figure 13.11 Dinsmore 1490 Digital Compass Sensor.

- **Electronic Compass Sensors**

  Low-cost electronic compass modules are also available that detect the magnetic field of the earth to indicate direction. The cost is two to three times that of the mechanical compass described in the previous section. New generation electronic compass modules offer more accuracy and faster settling times than mechanical compass sensors. An electronic compass module from PNI is shown in Fig. 13.12. This module contains a 2-axis magneto-inductive sensor and an ASIC. Heading information and magnetic field measurement data is available using a digital SPI serial interface.

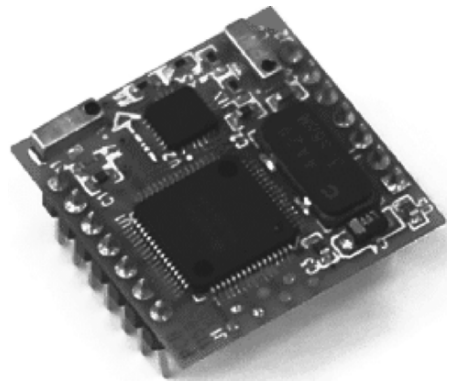


Figure 13.12 PNI Electronic Compass Module.

- **Low-cost Gyros and Accelerometers**

  Gyros and accelerometers are useful sensors for robots that need a balance sense. This can include robots that balance on two wheels like the Segway Human Transporter, robots that walk on two legs, and even robots that fly. Gyros and accelerometers have traditionally been used in aircraft autopilots and inertial measurement units (IMUs). Helicopters use a gyro to stabilize and control the tail rotor. Recently, Microelectromechanical Systems (MEMS) technology has produced small low-cost piezo-gyroscope and accelerometer ICs. These devices were originally used in automobile airbags. The gyros output a voltage level that is proportional to the speed or rate of the tilt angle changes. An analog-to-digital converter will be needed to input the gyro signal. The MEMS accelerometers output a voltage level or a pulse that changes its

duty cycle proportionally (e.g., PWM) to the tilt angle by sensing the change in acceleration due to gravity. Gyros will drift slowly over time and an accelerometer is needed to correct the gyro's drift. Without an accelerometer to correct for gyro drift, the tilt error slowly grows to the point where the robot would lose its balance. Accelerometers will respond more slowly to tilt than the gyro, so both a gyro and accelerometer is typically needed for each axis that needs a balance sense.

A complementary filter is used to combine or fuse sensor data from both the gyro and accelerometer to generate a more accurate tilt angle. Kalman filtering techniques can be used to improve the accuracy of noisy measurements. Noise levels are still somewhat high at very low G forces on these low-cost gyros and accelerometer IC sensors, so currently they are not useful for navigation since they cannot accurately determine the exact location of a slow moving robot by integrating the sensor measurements over time.

Analog Devices makes a variety of these sensors and sells small evaluation boards for them. It is likely that small low-cost sensor modules containing both a MEMS gyro and an accelerometer with a microcontroller will be available commercially in the near term.

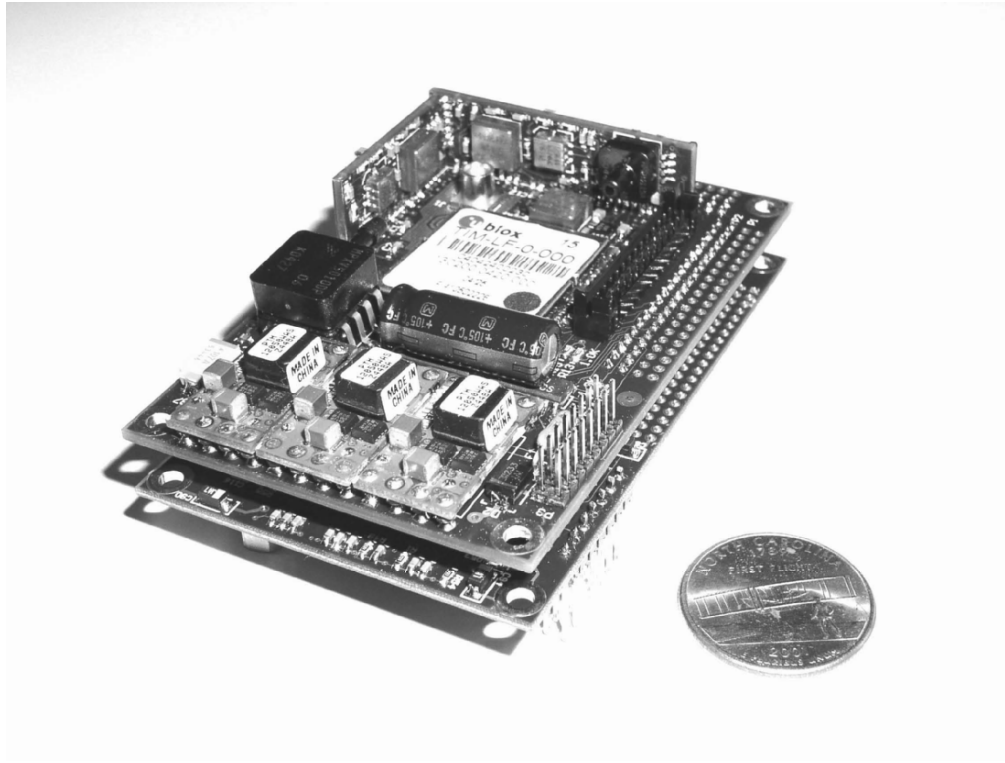


Figure 13.13 Small sensor board for an aircraft autopilot system. Photograph ©2004 courtesy of Henrik Christophersen , Georgia Institute of Technology Unmanned Aerial Research Facility.

Figure 13.13 shows a sensor board for an autopilot system that is used for unmanned aircraft. In the top corner, three MEMS gyros and accelerometers are mounted at right angles to provide data on all three axes. The white square flat module between the two vertical assemblies is a GPS receiver. The black square ICs at each end of the vertical assemblies are airspeed and altitude sensors. An A/D chip with an SPI interface is used to read sensors that have analog voltage outputs. The three square modules near the bottom edge of the board are DC to DC voltage converters. The lower board contains an FPGA and a DSP processor.

- **GPS and DGPS receivers**

The Global Positioning System was built by the US Department of Defense to provide highly precise worldwide positioning. Triangulation using radio signals from several satellites provides a position accurate to 25 meters. With an additional land-based correction signal, Differential GPS (DGPS) improves the accuracy to 3 meters. DGPS receivers provide ideal position data for robot navigation. Unfortunately, with current systems you are not likely to receive the GPS radio signals indoors in most buildings, so their use is typically limited to larger more rugged outdoor robots. Low-cost single chip GPS modules such as the Motorola FS Oncore seen in Figure 13.14 or the Ublox in Figure 13.13 are currently available. An SPI serial interface is supported. A new generation of highly sensitive GPS systems is being developed that may function indoors in some buildings.

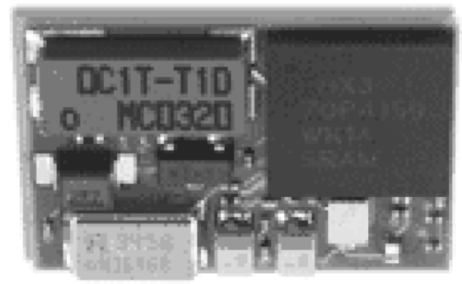


Figure 13.14 Motorola Single Chip GPS module.

- **Thermal Image Sensors**

Low-cost thermal image sensors can provide thermal imaging data for robots. Most thermal sensors such as those used in motion detectors and burglar alarms detect only movement. Thermopile sensors measure the temperature of a heat source. One such sensor, the Devantech TPA81 Thermopile Array is shown in Figure 13.15. It contains 8 Pyro-electric sensors arranged in a column that

detect infra-red in the radiant heat range of 2um to 22um range. It contains an on-board PIC microcontroller.

When the sensor is mounted on a servo, it can be used to horizontally scan an area and generate a thermal image. Candle flames and human body heat can be detected several feet away at room temperature. It uses an I^2C bus for interfacing to the host controller.

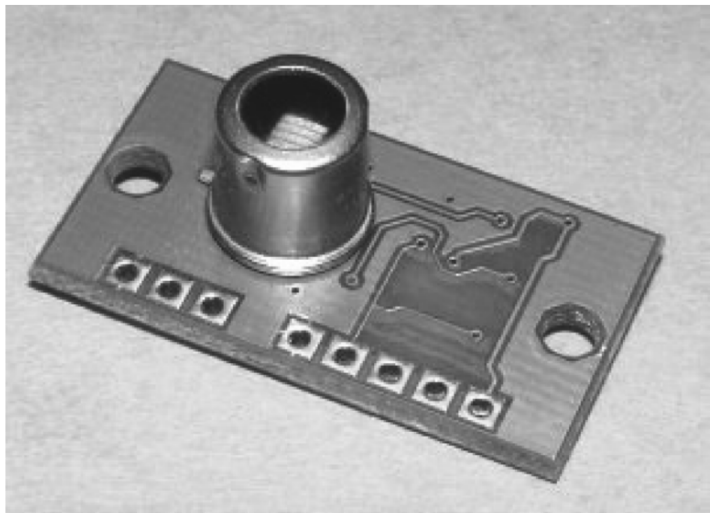


Figure 13.15 Devantech TPA81 Eight Pixel Thermal Array Sensor.

- **Solid State Cameras**

Low-cost solid state cameras can provide visual sensors for robots. Keep in mind that advanced image processing and visual pattern recognition requires complex algorithms that need a lot of processing power. The CMUCAM2 developed at Carnegie Mellon University seen in Figure 13.16 contains a PIC microcontroller and can transfer image data using a serial connection. It can track color blobs and report their location and size in an image at 26 to 50 frames per second.

Low-cost USB cameras are another option, but they will require a USB core interface and additional image processing. The low-cost CMOS color camera assembly OV6620 or OV7620 used in the CMUCAM2 module from Omnivision (www.ovt.com) can also be directly interfaced to an FPGA. It uses an I^2C interface for camera control signals and a separate parallel bus is used to transfer image data.

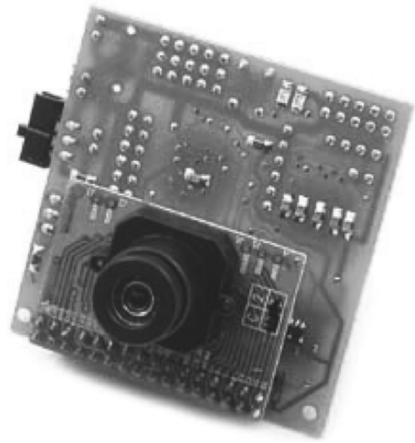


Figure 13.16 The CMUCAM2 contains a color video camera on a chip and a microcontroller.

13.6 Assembly of the FPGA-bot Body

Assembly of the FPGA-bot can be accomplished in about an hour. A drill or drill press, screwdriver, scissors, a soldering iron, and a wire stripper are the only tools required. First, obtain the parts in the parts list. Next, drill out the holes in the round Plexiglas base (part #15) as shown in Figure 13.17. The mounting holes will move around depending on which FPGA board is used. The DE2 board is a bit larger than the other boards and you may want to increase the size of the plastic base to accommodate it or mount it on long standoffs well above the servo wheels.

To prevent scratches, leave the paper covering on the Plexiglas until all of the holes are marked and drilled out. The front of the base is on the right side in Figure 13.17. The wheel slots are symmetric with respect to the center of the circle.

Proper alignment of the four screw mounting holes for the FPGA board is critical. Unscrew the four standoffs from the bottom of the FPGA board. Carefully place it towards the rear of the plastic base as shown in Figure 13.17, and mark the location of the screw holes using a pen or pencil. Any FPGA board can also be used, but the mounting holes will be in different locations. Leave extra space in front of the FPGA board on the plastic base for use by forward facing sensor modules as seen in Figures 13.1 and 13.9. Double check that the board clears the top of the servo wheels once it is mounted. Longer standoffs can also be used to clear the wheels, if needed. Locate the cable and switch holes as shown in Figure 13.17. Exact positioning on these holes is not critical. If one is available, use an automatic center punch to help align the drill holes. The board's expansion header pins should face towards the front of the

base so that it is easy to attach the sensors. Re-attach the standoffs to the FPGA board and set it aside. After all holes are drilled, remove the paper covering the Plexiglas.

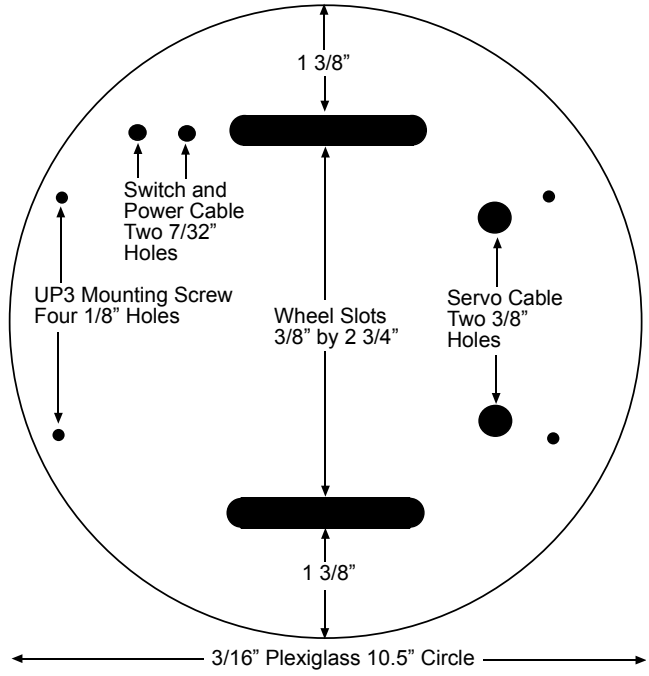


Figure 13.17 FPGA-bot Plexiglas Base with wheel slots and approximate drill hole locations for the UP3 board. Verify the exact dimensions and mounting hole locations for each individual FPGA board.

Mount the toggle switch (part #8) in the hole provided in the base. If available, Loctite or CA glue can be used on the switch mounting threads to prevent the switch nut from working lose. Solder the red wire (+7.2V) from the battery connector to one of the switch contacts. This is the connector with wires that plugs into the battery pack connector (part #4). Solder one of the twin lead wires (part #9) to the other switch terminal. Solder the other twin lead wire to the black (GND) battery connector wire and insulate the splice with heat shrink tubing or electrical tape (part #10).

Route the twin lead wire through the hole provided in the base. A small knot in the twin lead on the bottom side of the base can be used for strain relief. Solder the power connector (part #11) to the other end of the twin lead wire on the top of the base. The center conductor is +7.2V and the outer conductor is ground on the power connector.

Check the power connections with an ohmmeter for shorts and proper polarity before connecting the battery. For strain relief and extra insulation, consider sealing up the power connector with Silicone RTV or insulating one of the wire connections with heat shrink tubing. Be careful, NiCAD and NiMH batteries have been known to explode or catch on fire, if there is a short. A fuse on the battery power wire might be a good idea, if you are prone to shorting out circuits.

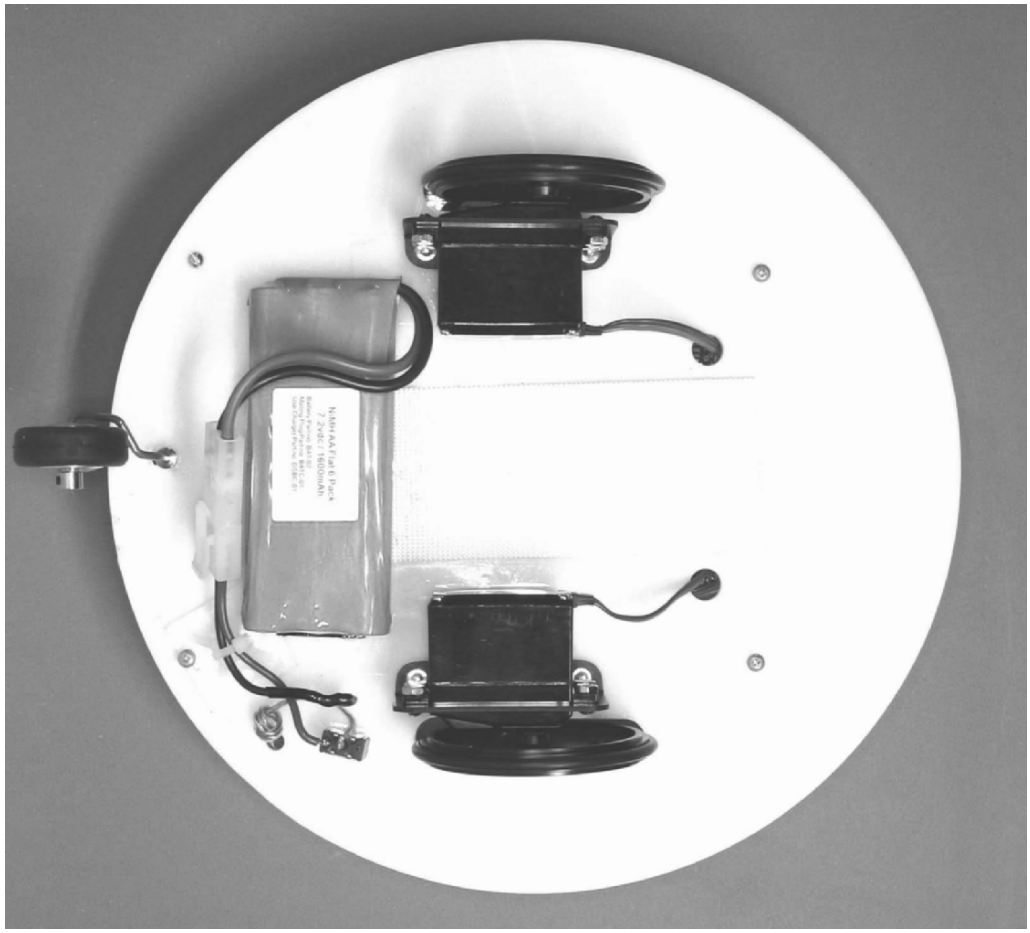


Figure 13.18 Bottom view of FPGA-bot base showing battery, servos, wheels, and cabling.

Attach the battery pack (part #2) to the bottom of the Plexiglas base with sticky-back Velcro (part #16). Figure 13.18 is a close-up photo of the bottom side of the FPGA-bot. A NiMH battery pack is shown in Figure 13.18. If you use a larger NiCAD battery pack, it can be mounted in the middle of the base about one inch off center towards the rear wheel, with the battery pack connector facing the rear.

The battery is moved towards the rear for balance to place the weight on the rear skid. The Velcro on the base should be around 2 inches longer than the battery pack towards the rear of the robot to allow for positioning of the battery later on to balance the robot. The wire and connector on the battery pack should also be attached to the base to prevent it from dragging on the floor. Attach a small piece of Velcro on the rear of the connector so that the battery wires can be attached to the base. Attach the battery pack to the base.

On UP3, UP2, and UP1 boards, solder a 60-pin female header socket (part #7) to the expansion header B location. Attach the FPGA board to the top of the base with 4-40 screws (part #18), using the hex spacers provided on the FPGA board (part #14). Figure 13.19 is a close-up photo of the top of the FPGA-bot. Double check power connections and polarity with an ohmmeter. The inner contact on the power connector should be +7.2V, the outer contact is ground, and the toggle switch should turn it off. Then plug the power connector into the FPGA board. Plug in the battery connector and flip the power switch. An LED should light up on the FPGA board indicating power on. The Cyclone expansion B header socket faces the front of the robot. DE2 and DE1 boards already have expansion sockets soldered to the board.

Mount the wheels (part #5) on two modified servos (part #3). If you are not using the special servo wheels, you may need to enlarge the hole in the center of each wheel by drilling it out partially with a drill bit that is the same size as the servo output shaft. The depth of the hole should be slightly shorter than the servo output shaft and not all the way through the wheel, so that the wheel does not contact the servo body. The servo output shaft screw is inserted on the side of the wheel with the smaller hole. A washer may be required on the servo screw. The wheel should not contact the servo case and must be mounted so that it is straight on the servo. CA glue or Blue Loctite can also be used to attach the wheels and screws more securely to the servo output shaft.

Attach the servos to the bottom of the base using double sided foam tape (part #17) or a more durable servo mounting bracket. The servo body faces toward the center of the base. Be sure to carefully center the wheels in the plastic-base wheel slot. If you are using foam tape, make sure all surfaces are clean and free of grease, so that the foam tape adhesive will work properly. Lightly sanding the servo case and adding a drop of CA glue helps with tape adhesion. Route the servo connector and wire through the holes provided in the base.

Attach a tail wheel to the base or a skid (part #19) at the rear of the battery pack using layers of foam tape as needed. Move the battery as needed so that the robot has proper balance and rests on the two wheels and the rear skid.

Attach another skid to the front of the battery pack using several layers of foam tape. The front skid should not contact the floor and at least ¼ inch of clearance is recommended. The front skid only serves to prevent the robot from tipping forward during abrupt stops.

On the UP3, attach a 3-pin .1 inch header (part #9) to the small wire wrap protoboard in an open area. One is required for each servo on the robot. Solder wires from the appropriate pins J2 and J3 connections on the protoboard to the new header pins. The three wires on the servo are Vcc (4.8 to 6 volts), ground,

and the PCM control signal wire. Some manufacturers' servos have different power connections, but they all have three pins. Wrap extra servo wire around the hex spacers underneath the UP3 board.

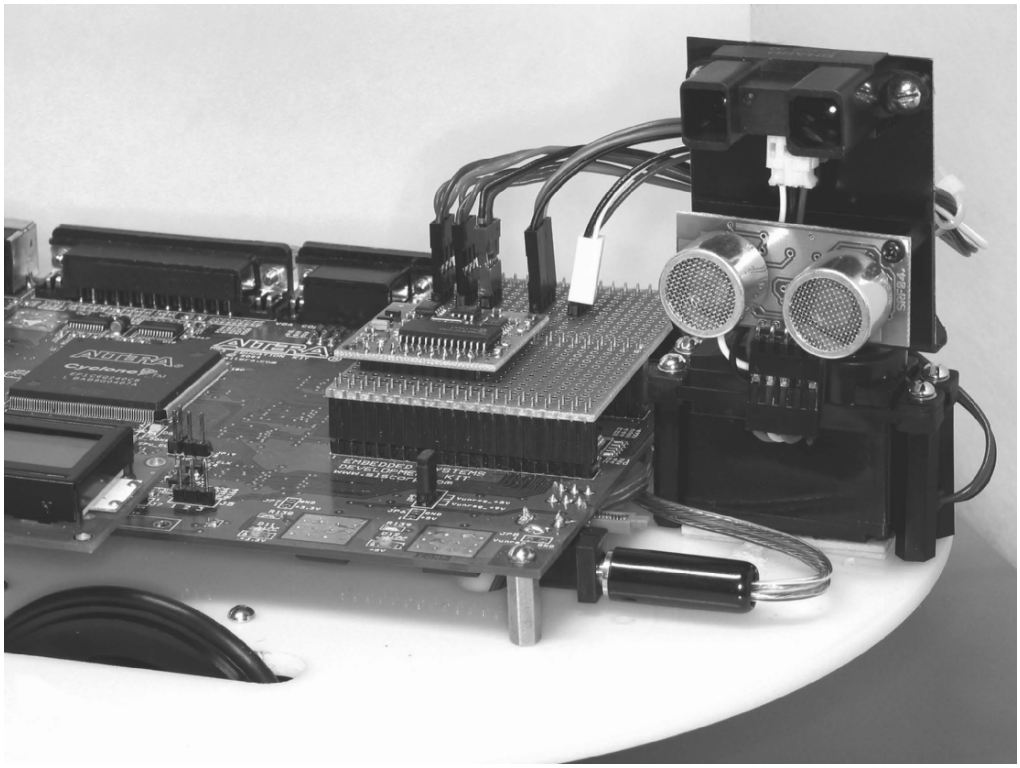


Figure 13.19 Top View of FPGA-bot Base with Compass, IR, and Sonar Sensor Modules. A UP3 FPGA board is shown, but any FPGA board can be used.

Optional sensor modules such as the IR proximity detector or line tracker can be attached to the base unit with foam tape. Run wires from the sensors to the expansion connectors. A small .1 inch wire wrap protoboard with 40-pin female header connectors soldered to the protoboard as shown in Figure 13.19 is handy for making servo and sensor connections to the FPGA board. In Figure 13.19, a third servo is used to make a sensor turret for IR and Sonar distance sensors.

- **Parts List for the FPGA-bot**

1. **An Altera FPGA Board**. The FPGA board serves as the controller for the FPGA-bot. It is attached to the FPGA-bot body with screws. No modifications are required to the board. Any of the Altera FPGA boards can be used, but mounting holes will be in different locations.

- **Parts Available from a Hobby Store**

2. **A 7.2V to 8.4V 1300-1700mAh Rechargeable NiCAD battery pack with the standard Kyosho battery connector.** This is a standard R/C car part, and it is used to power the FPGA-bot. For a small additional cost, new NiMH batteries are also available that store almost twice the energy per weight. The battery will need to be charged prior to first use.

3. **Two modified R/C Servomotors**. Two identical model servos are required so that the motors run at the same speed. Servo modifications are described in section 13.3. Any servo should work. The following servos have been tested: Tower Hobbies TS53J, Futaba S148 and S3003, and HS 300. Some manufacturers' servos appear to run in the reverse direction. This is easily fixed in the hardware design since the motor controller is implemented on the UP3 board. Several robot parts vendor sell modified servos for a slightly higher cost. Ball bearing servos are worth the extra cost, if you intend to run the robot constantly for several months. A third unmodified servo will be needed if you want a rotating sensor turret as shown on the example FPGA-bot photo.

4. **Kyosho Female Battery connector with wire leads, Duratrax or Tower Hobbies #DTXC2280**. This is used to connect to battery. A connector is needed so that the battery can be disconnected from the FPGA-bot and connected to a charger.

5. **Two Acroname or Lynxmotion servo wheels.** These wheels are 2 ¾ plastic wheels that are designed to attach to the servo's output shaft spline. Prather Products 2¼-inch aluminum racing wheels with rubber O ring tires, Tower Hobbies #PRAQ1810 or Hayes Products #114, 2 ¼-inch hard plastic racing wheels (also available from Tower Hobbies) can be used as a substitute. These somewhat smaller two alternative wheels will work, but they do not have the spline to match the servo output shaft and are a bit more difficult to connect reliably than the Acroname or Lynxmotion servo wheels.

6. **A Castering Wheel or Two small Teflon or Nylon Furniture Slides.** There is a bit too much mechanical play in common furniture casters for a small robot and they tend to randomly deflect the robots direction after sharp turns. Lynxmotion's #TWA-01 is a mini castering robot tail wheel built using an R/C airplane tail wheel that works well. The mounting wire needs to be bent a little off center so that the wheel quickly rotates to the direction of travel. Other robot parts vendors such as Acroname also have robot tail wheels, but a spacer may be required to adjust the height. The battery will need to be moved a bit and perhaps rotated ninety degrees to accommodate them and still maintain proper balance on the robot base. Magic Sliders 7/8-inch diameter circular discs also work well on flat surfaces. The slides are used as a skid instead of a third wheel on the FPGA-bot. Metal or hard plastic will also work. Attached to the bottom of the battery with several layers of foam tape, an optional front skid can be used for stability during abrupt stops. On flat surfaces, a Teflon skid actually works better than a common small furniture caster from a hardware store.

7. **A charger for the 7.2V or 8.4V battery pack**. An adjustable DC power supply can be used to charge the battery if it is properly adjusted and timed so that the battery is not overcharged. Overcharged batteries will get hot and will have a shorter life. Automatic peak-detection quick chargers are the easiest and most foolproof to use. These chargers shut off automatically when the battery is charged. One quick charger can be used for several robots as a full charge is achieved in less than 30 minutes with around 5 Amps maximum charge current. Inexpensive trickle battery chargers deliver only around 75 mA of charge current, and they will require several hours charge the battery.

- **Parts Available from an Electronics Parts Store**

8. **Three 40-pin .1-inch double row PC board mount female header sockets**, **DigiKey #S4310 or equivalent**. These sockets are soldered into a small 0.1" center wire wrap protoboard that fits into the Santa Cruz Expansion connector on the UP3. This is used to connect servos and sensors to the UP3 board.

9. **A 2 to 3 inch strip of .1" single row breakaway headers. DigiKey #S1021-36 or equivalent** These headers are used to make custom servo and sensor connectors on the protoboard. They can be soldered to the protoboard.

10. **A small wire wrap protoboard with holes on .1" centers cut down to 2" by 2.8". A** This is used to make a protoboard for use with the UP3 board. The protoboard contains connectors for servos and sensor. A protoboard with solder pads makes it easier to mount the connectors.

11. **A miniature toggle switch with solder lug connections.** The switch should have a contact rating of more than two amps (Radio Shack #275-635B or equivalent). Only two contacts or single pole single throw (SPST) is needed on the switch to turn power on and off. If all of your servos and sensors connect to the UP3 and do not use the Vunregulated supply, you could eliminate the switch by using the UP3's power switch.

12. **Approximately 9 inches of small-gauge twin-lead speaker wire.** This part is used to connect power to the UP3 board. The wire must fit into the DC power plug (part# 11). Typically, 20-22 gauge wire is required. Two individual wires can also be used, but twin lead is preferred.

13. **A 1-inch piece of small heat shrink tubing or electrical tape.** This part is used to insulate a splice in the twin-lead power wire.

14. **A Coaxial DC Power Plug with 5mm O.D. and 2.1mm I.D., Radio Shack Number 274-1567** or equivalent. This power plug fits the power socket on the UP3 board. A different size plug is needed for the UP2 board, use #274-1568 that has a 2.5mm I.D.

15. **An assortment of small wire jumpers and connectors to attach wires to the male headers on the UP3**. These are the jumper wires commonly used for protoboards. Two

short jumpers are used to connect the two servo signal wires, and other jumpers are used to any connect sensor boards.

16. **Four, 1-inch hex spacers with 4-40 threads or use the shorter spacers that come with the board.** These are used to mount the UP3 board to the Plexiglas base using the holes in the UP3 board.

- **Parts Available from a Hardware Store**

17. **3/16-inch thick Plexiglas cut into a 10.5-inch diameter circle**. This part is the base of the robot. Colored Plexiglas such as opaque white, will not show scratches as easy as clear. Holes to cutout and drill are shown in Figure 13.11. If a band saw, jig saw, or other machine tool is not available, a local plastics fabricator can cut this out. When using a number of very large sensors or a DE2 board, it may be necessary to increase the size slightly or add another circular deck for sensor mounting. A larger robot requires more space for maneuvering. To prevent scratches on the Plexiglas, keep the paper backing on the plastic until all of the holes have been marked and drilled out. The size of the wheel slots may need to change depending on the wheels you select.

18. **One 8-inch long strip of 2-inch wide sticky-back Velcro**. Two 8-inch long strips, 1 inch wide can also be used. The Velcro is used to attach the battery to the bottom of the Plexiglas base. Since the battery is attached with Velcro and a connector, it can be quickly replaced and removed for charging.

19. **Approximately 8 inches of 1-inch wide double-sided 3M foam tape**. This is used to attach servos, skids, and optional sensor boards to the base. Be sure to clean surfaces to remove any grease or oil prior to application of the tape for better adhesion. For a more durable servo mount, Lynxmotion has aluminum servo mounting brackets that can be used instead of the double sided tape.

20. **Four 4-40 Screws 5/16-inch or slightly longer**. The screws are used to attach the UP3 board to Plexiglas. The screws thread into the hex spacers attached to the UP3 board.

21. **Blue Loctite, Cyanoacrylate (CA) Glue, and Clear Silicone RTV.** These adhesives and glues are useful to secure screws, servos, and wheels. The mechanical vibration on moving robots tends to shake parts loose over time. These items can also be found at most hobby shops. Only a few drops are needed for a single robot. A single tube or container will build several robots.

13.7 I/O Connections to the board's Expansion Headers

Most servos and sensor I/O signals will need to be attached to an expansion header. Many FPGA pins are now 3.3V. R/C Servos and most sensors use 5V, but be sure to check the device's datasheet. Don't forget to connect a ground signal between the device and the FPGA board, even if the device has it's own power supply or a direct connection to a battery. Several ground and power pins

are available on the FPGA board's expansion headers. A 5V 1A power supply pin is available on the UP3's J2 expansion connector. J4 has a 3.3V supply connection pin and JP6 can be used as another 5V supply connection.

A small protoboard can be built to connect servos and sensors to the FPGA board. All of the connectors and pins on the boards line up on tenth inch centers. A 0.1" perfboard or wire wrap protoboard can be cut down to 2" by 2 7/8" so that it fits over J1, J2, J3, and J4. 0.1" 40 pin connectors to attached the protoboard to connect to J1..4 can be mounted on the protoboard. A wire warp protoboard with holes every .1" has solder pads that can be used to attach connectors using solder. Point to point wiring and soldering can be used to make connections on the protoboard from the J1..J4 connectors to the .1" connectors used to attach servos and sensors. Small single row strips of .1" header pins can be snapped apart to make male connectors on the board for the servos and most sensors.

You may want to consider isolating your robot's servo or motor power supply from the supply used for the FPGA board's logic to control the noise generated on the supply lines by the DC motors. On larger robots, two batteries are sometimes used. A $V_{unregulated}$ connection that does not go through the 5V regulator and is connected directly to the UP3's power input jack is available on JP8 and J4. The 9V supply is connected after the input power switch on the UP3 and to JP5. This also can be used to power servos and motors, assuming the battery voltage level is not too high. If the battery voltage is too high, another regulator can be used for the motors.

At a minimum, decoupling capacitors connected across the servo's power supply connections are a good idea. If you plan on having several sensors on your robot, you may want to consider building a small PCB with header pins for the sensor power and data connections as seen in Figure 13.19. Most R/C servos can run on 4.8 to 6V.

13.8 Robot Projects Based on R/C Toys, Models, and Robot Kits

A second option for building an FPGA driven robot involves modifying a low-cost radio-controlled (R/C) car or truck. Fundamentally, almost any large R/C car or truck can be modified to work with the Altera board, although some are clearly better choices than others.

In our robot, we used a Radio Shack (www.radioshack.com) R/C 4WD SUV shown in Figure 13.20. The R/C platform affords a more robust drive train and control; however, turning radius and noise levels are sacrificed over the smaller FPGA-bot. The R/C SUV has a spring suspension and large soft tires that make it operable outdoors on rougher surfaces. Following are some R/C car selection considerations that will affect available modifications and control of the new platform.

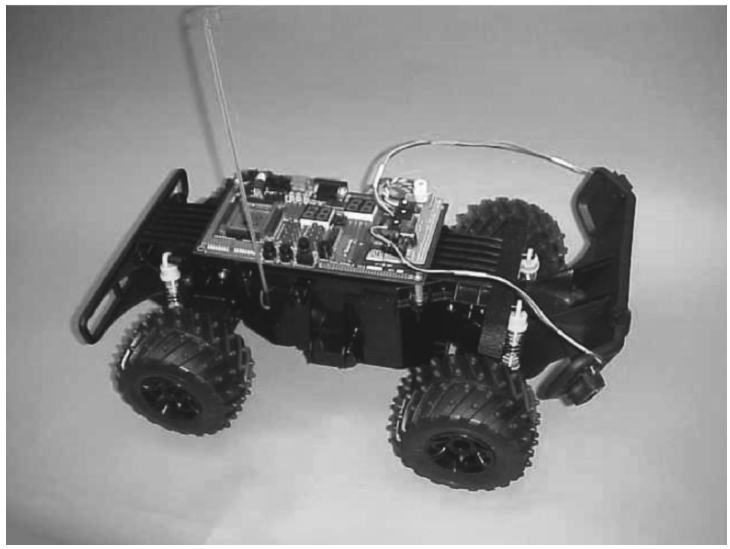
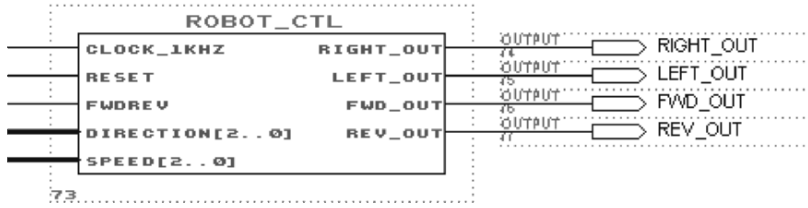


Figure 13.20 FPGA Controlled Toy R/C Truck with IR Distance Sensors.

- **Seven-Function Controls**

  When choosing an R/C car, select one that has a remote control with at least seven remote functions (forward, backward, forward-right, forward-left, backward-right, backward-left, and stop). Note that these low-cost R/C cars do not have variable speed or variable turning controls; however, once they are interfaced to the FPGA board, variable speed and turning can be accomplished by changing the duty cycle of the command signals. (More on this later.)

  A control module built using the FPGAs logic allows a relatively inexpensive R/C car to perform with the capabilities of the more expensive cars with "digital proportional steering" and "digital proportional speed controls." Once interfaced to the FPGA board, an IP core (Robot_CTL) is used to handle control of all direction and speed control outputs. As illustrated in Figure 13.21, the IP core control module affords a higher degree of control than the original radio control. The outputs connect to the R/C cars internal control circuits that drive the DC Motors.



FwdRev	1 Bit	0 = Forward/1 = Reverse
Direction	3 Bits	First bit Left/Right, 2nd and 3rd bit is angle.
		0-00 = Left – Straight*
		0-01 = Left – Slight Turn
		0-10 = Left – Medium Turn
		0-11 = Left – Full Turn
		1-00 = Right – Straight*
		1-01 = Right – Slight Turn
		1-10 = Right – Medium Turn
		1-11 = Right – Full Turn
		* Note: 000 and 100 are both Straight
Speed	3 Bits	000 = Stop
		001 = Slowest Speed
		: : :
		111 = Fastest Speed

Figure 13.21 Robot Control IP Core with Pulsed Speed & Steering Control.

- **Speed**

  When considering the speed of the vehicle, a modest speed is more desirable than the faster speeds. At 800 feet per minute, our prototype FPGA controlled robot car moves fast enough to be difficult to catch. In almost all cases, the robot is operated at half the maximum speed or less. The limiting factor is generally the delay inherent in the sensor's input sampling rate and range. A fast moving car typically will hit the wall before a collision sensor can take the data samples needed to initiate avoidance.

  To control the speed (or the degree of turn) a repeated pulse train is sent to the forward or reverse signals (left or right signals for direction). Instead of a steady high signal causing the car to move forward at full speed, the pulse train varies the duty cycle to change speed (or degree of turn). By modulating the duty cycle of the pulse train, "digital proportional control" can be implemented on each control signal. In other words, changing the duty cycle can control the speed and the degree of turn. The more the duty cycle approaches 100%, the harder the turn and/or the faster the speed.

The frequency of the pulsed control signal used must be higher than the natural mechanical frequency response of the system. A very slow changing pulse will cause the motor and gears to vibrate and make additional noise. Pulse frequencies of a few kHz are typically used to avoid this problem.

When reversing direction on a moving DC motor, it is common practice to include a small time delay with the motor turned off to reduce the inductive voltage spikes produced by the motor windings. Recall that changing current flow through an inductor produces voltage. Without the delay in some circuits, these high voltage spikes can damage or reduce the life of the transistors controlling the motor. This delay can be incorporated in the IP control core, if needed.

Figure 13.22 illustrates the relationship between turn angle, speed, and duty cycle on the control signals. The figure implies that the duty cycle is linear, i.e., a 50% duty cycle produces half speed. Actually, the duty cycle is very non-linear and highly dependent on the type of car, size of the DC motors, and power. (An R/C car with a dying battery performs as if the duty cycle is considerably less.) By experimenting with patterns of the 16-bit speed and direction vectors used in the IP core controller, a more linear relationship can be established between the command bit patterns and the actual performance of the vehicle.

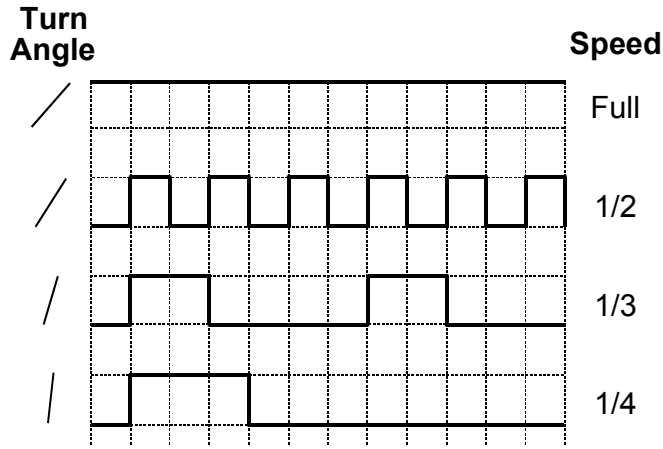


Figure 13.22 Affect of Duty Cycle on Turning Angle and Speed**.**

- **Battery Choice**

The choice of car will also dictate the type of batteries and charger that will be needed. Note that some cars come with 9.6V packs and others come with 7.2V packs. Both should work well with the UP3 board as a controller. The prototype used the 7.2V pack that discharged quickly and required a second pack placed in parallel with the first to support longer run times. The cars with a 9.6V pack should give the UP3 board's 5V onboard regulator a better regulator margin and a longer life between recharges.

- **Mounting the FPGA Board**

  Before you select an R/C car, make sure that there is a good place to mount the FPGA board. If the car is large enough, there is usually a large flat area under the car body cover molding to secure the FPGA board.

- **Interfacing the FPGA Board to the R/C Car**

  Remove appropriate body cover screws and expose the PC board receiver and control module. Most current low-cost R/C toy cars have a single electronic PC board that contains both control circuits. Generally, there is one 16 or 18-pin DIP radio command demodulator chip in the center of the board that converts the radio signals into simple digital control signals. These digital control signals then activate the H-bridge circuit that controls the DC motors that drive the wheels of the R/C car.

  An H-bridge is a standard electronic circuit used to control DC motors. It allows for both forward and reverse operation of the same DC motor. H-bridge circuits contain four large power transistors that are needed to turn on and reverse a DC motor. Discrete transistors may be used to build the H-bridge or it may be in an IC or packaged module that connects directly to the motors.

NEVER ATTEMPT TO DRIVE A MOTOR OR RELAY WITH AN FPGA PIN DIRECTLY, IT CANNOT SUPPLY THE HIGH CURRENT LEVELS THAT THESE DEVICES NEED. THE FPGA'S OUTPUT PIN DRIVER CIRCUIT MAY BE DESTROYED.

  If the car supports seven functions, it will have at least four pins coming off of the DIP chip package that break down into Left, Right, Forward, and Reverse. Using a voltmeter or an oscilloscope, test which pins change when the remote control is set to each of the four directions. From each of the designated command pins on the chip, the trace on the PC board will run to separate H-bridge circuits for each motor.

  By clipping or desoldering and pulling out the four control pins on the chip going to the board and soldering wires from each chip pin hole pad trace to the FPGA (Figures 13.23 and 13.24), the FPGA board can control the four directions and speed of the R/C car using the car's existing H-bridge circuits. In our modification, we desoldered the entire chip and put a socket on the board. To have the original control signals coming from the radio control module also sent to the FPGA board, run four more wires from the clipped chip pins to the one of the FPGA's headers. If the four clipped chip pins (RC*x*) are connected to the FPGA board, logic can be designed to include the original radio control functions supplied by the handheld remote control unit. You can also make this connection at the H-bridge circuit if necessary. On the UP3, you will want to use the UP3s 5V I/O pins for this interface.

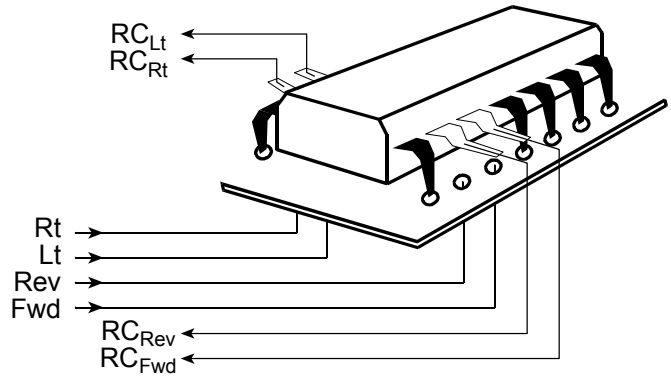


Figure 13.23 Interfacing to the R/C Car's Internal Control Signals at the Demodulator IC.

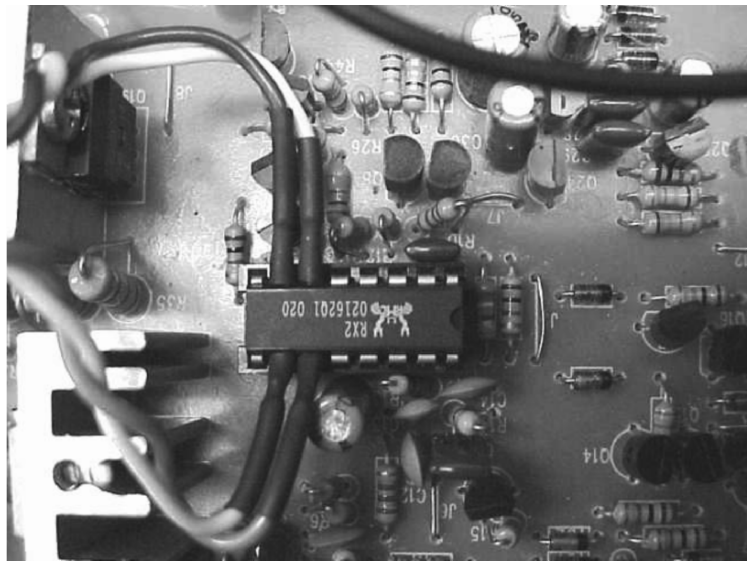


Figure 13.24 Photo Showing Control Modifications to R/C Car Control Board.

- **Hobbyist R/C Models, Robot Kits, and Commercial Robot Bases**

  For those with a larger budget, higher quality R/C hobbyist cars are available with built-in proportional steering and pulsed electronic speed controls. The control interface to these cars uses standard R/C PWM signals that are identical to the PWM servo control bit described at the end of Section 13.2. An example R/C Hummer can be seen in Figure 13.25. This robot is controlled using a C program running on the FPGA's Nios processor core. Various robot kits without control electronics or a computer are also available. Almost any of these robot kits can be controlled by the UP2 or UP3 board provided they are large enough to carry it and can power it from their battery or carry a second battery for the

UP3. An interesting walking robot kit containing 12 R/C servos is seen in Figure 13.26.

Figure 13.25 Hobbyist R/C model with a CMU camera and R/C PWM servos controlled by an FPGA

Figure 13.26 Lynxmotion Hexpod Walking Robot Kit with 12 R/C servos

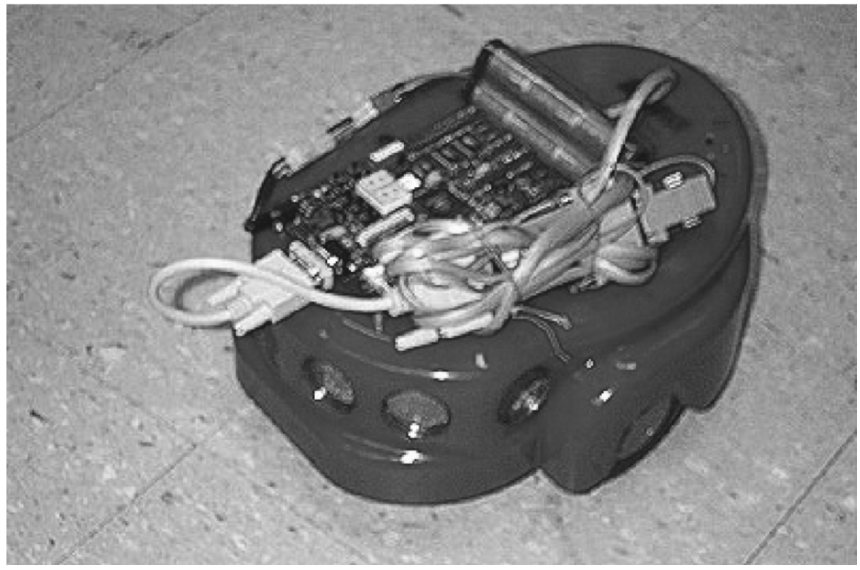


Figure 13.27 ActiveMedia's Amigobot robot base controlled by an FPGA with a Nios Processor

Some commercial robot bases are also available such as the Amigobot as seen in Figure 13.27, the ER1 from Evolution Robotics and the mobile robot platform from Drrobot. The Amigobot uses an RS-232C serial interface and the ER1 uses USB for motor control. A robot base contains a motor, drive electronics, sensors, and a battery, but it has no high-level controller.

One of the newest and most attractive low-cost options for a robot base is iRobot's iCreate robot base as seen in Figure 13.28 (www.irobot.com). It is basically a "Roomba" robotic vacuum cleaner without the vacuum parts. It contains an internal microcontroller, two drive motors with feedback, audio output, and several simple sensors (i.e., IR, bump or contact switches, and IR cliff or surface drop-off sensors). Using an RS-232 serial port, motor commands can be sent to the microcontroller and the sensor status can be read back. An FPGA board running a Nios processor can talk to the iCreate microcontroller using the RS-232 serial port. You can write the required robot application code for the Nios processor in C.

6-32 screw holes are provided on top of the base that can easily be used to mount an FPGA board using an additional flat rectangular piece of Plexiglas or Lexan. The rectangular sheet of plastic bolts to the base using the four screw holes above the open cargo area, and the FPGA board bolts to the sheet of plastic. Power can be provided at up to 1 amp using the iCreate's internal battery or there is also space in the cargo area for an additional battery pack for the FPGA board. The internal battery voltage is 18V, so a 7, 9, or 5 volt DC regulator (depends on which FPGA board) is needed to drop the voltage for the FPGA board (note: too high of a DC input voltage will overheat the FPGA board's internal DC regulator). A DB-25 connector seen at the center in Figure 13.28 provides all of the power and serial port connections needed.

Figure 13.28 iRobots iCreate robot base can be controlled by an FPGA with a Nios Processor using an RS-232 serial port to send commands to the iCreate robot's internal microcontroller. Photograph courtesy of iRobot.

Once you develop a working robot and want to run existing demos, you may want to program the FPGA board's flash memory configuration device so that your design automatically runs whenever the board is turned on.

13.9 For Additional Information

Radio-controlled cars and parts such as batteries, battery chargers, and servos can be obtained at a local hobby shop or via mail order at a lower cost from:

Tower Hobbies http://www.towerhobbies.com
P.O. Box 9078
Champaign, IL 61826-9078
800-637-6050

IR Proximity, Line Tracker, Sonar sensors, Servos, Wheels, and Robot kits can be obtained via mail order from:

Lynxmotion, Inc. http://www.lynxmotion.com
104 Partridge Road
Pekin, IL 61554-1403
309-382-1254

Mondotronics http://www.robotstore.com
4286 Redwood Highway #226
San Raphael, CA 94903
800–374–5764

Sensors & Robot kits, Servo wheels for robots, and Servo wheel encoder kits can also be obtained via mail order from:

Acroname http://www.acroname.com
P.O. Box 1894
Nederland, CO 80466
303-258-3161

Low-cost digital and analog compass sensors are available via mail order from:

Dinsmore Instrument Co. http://www.dinsmoresensors.com
P.O. Box 345
Flint, Michigan 48501
810-744-1790

Electronic Compass Modules are available from:

PNI Corp http://www.pnicorp.com
5464 Skylane Blvd. Suite A
Santa Rosa, CA 95403

GPS and DGPS ICs and modules are available from:

Motorola TCG http://www.motorola.com/gps
GPS Products
2900 South Diablo Way
Tempe, AZ 85282

u-blox AG http://www.u-blox.com
Zürcherstrasse 68
8800 Thalwil
Schweiz

A wide array of robot sensor modules is available from:

Devantech Ltd (Robot Electronics) http://www.robot-electronics.co.uk
Unit 2B Gilray Road
Diss
Norfolk
IP22 4EU
England

Robotics Connection
4355 Cobb Parkway
Suite J148
Atlanta, GA 30339
http://www.roboticsconnection.com

A wide array of various sensor and GPS boards is available from:

Spark Fun Electronics
2500 Central Ave.
Suite Q
Boulder, CO 80301
http://www.sparkfun.com

A longer list of robot parts vendors and sites can be found at:

http://users.ece.gatech.edu/~hamblen/4006/robot_links.htm

13.10 Laboratory Exercises

1. Develop a counter design to find the dead zone of a converted R/C servo motor. The dead or null zone is the time near 1.5ms that actually makes the servo motor stop moving. As in the example motor driver code, send a width adjusted pulse every 20ms. You will need a resolution of at least .01ms to find the dead zone, so a clock faster than the example code is required. For example, the motor might actually stop at 1.54ms instead of 1.50ms. Use the clk_div FPGAcore function to provide the clock. The design should increase the width of the timing pulse if one pushbutton is hit and decrease the width if the other pushbutton is hit. Display the width of the timing pulse in the seven-segment LEDs. Use a Cyclone DIP-switch input to select the motor to examine. By hitting the pushbuttons, you should be able to stop and reverse the motor. The dead zone will be between the settings where the drive wheel reverses direction. At the dead zone, the drive wheel should stop. Settings near the dead zone will make the motor run slower. Record the dead zone for both the left and right motor.
2. Using the dead zone settings from problem 1, design a motor speed controller. Settings within around .2ms of the dead zone will make the motor run slower. The closer to the dead zone the slower the motor will run. Include at least four speed settings for each motor. See if you can get the robot to move in a straight line at a slow speed.
3. Develop a speed controller for the robot drive motors by pulsing the drive motors on and off. The motors are sent a pulse of 1ms for reverse and 2ms for forward at full speed. If no pulse is sent for 20ms, the motor stops. If a motor is sent a 1 or 2ms pulse followed by no pulse in a repeating pattern, it will move slower. To move even slower use pulse, no pulse, no pulse in a repeating pattern. To move faster use pulse, no pulse, pulse in a repeating pattern. Using this approach, develop a speed controller for the robot with at least five speeds and direction. Send no pulse for the stop speed. Some additional mechanical noise will result from pulsing the motors at slow speeds. See if the robot will move in a straight line at a slow speed.
4. Use an IR LED and IR sensor to add position feedback to the motors. You can build it yourself or a similar servo wheel encoder kit built by Nubotics is available from Acroname. Some sensor modules are available that have both the IR LED and IR sensor mounted in a single plastic case. For reflective sensors, mark the wheels with radial black

paint stripes or black drafting tape and count the pulses from the IR sensor to determine movement of the wheel. Another option would be to draw the radial stripes using a PC drawing program and print it on clear adhesive labels made for laser printers. The labels could then be placed on the flat side of the wheel. If a transmissive sensor arrangement is used, holes can be drilled in the main wheel or a second smaller slotted wheel could be attached to the servo output shaft that periodically interrupts the IR light beam from the LED to the sensor. In this case, the LED and sensor are mounted on opposite sides of the wheel. This same optical sensing technique is used in many mice to detect movement of the mouse ball. Use the position feedback to implement more accurate variable speed and position control for the motors.

5. Design a state machine using a counter/timer that will move the robot in the following fixed pattern:
   - Move forward for 6 seconds.
   - Turn right and go forward for 4 seconds (do not count the time it takes to turn).
   - Turn left and go forward for 2 seconds.
   - Stop, pause for 2 seconds, turn 180 degrees, and start over.

   Determine the amount of time required for 90- and 180-degree turns by trial and error. A 10Hz or 100Hz clock should be used for the timer. Use the clk_div FPGAcore to divide the UP3 on-board clock. The state machine should check the timer to see if the correct amount of time has elapsed before moving to the next state in the path. The timer is reset when moving to a new portion of the path. Use an initial state that turns off the motors until a pushbutton is hit, so that it is easier to control the robot during download. Since there is no motor position feedback, all turns and the actual distance traveled by the FPGA-bot will vary slightly.

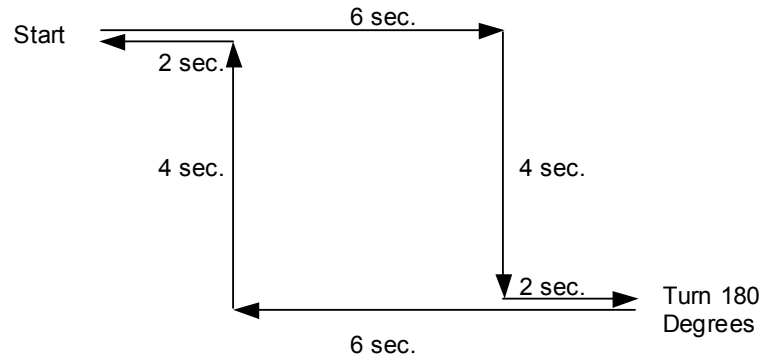


Figure 13.29 Simple path for state machine without sensor response.

6. Using a ROM, develop a ROM-based state machine that reads a motor direction and time from the ROM. Put a complex pattern such as a dance step in the ROM using a MIF file. For looping, another field in the ROM can be used to specify a jump to a different next address.

7. Using the keyboard FPGAcore, design an interface to the keyboard that allows the keyboard to be used as a remote control device to move the robot. Pick at least five different keys to command to robot to move, turn left, turn right, or stop.
8. Interface an IR proximity sensor module to the FPGA-bot using jumpers connected to the Cyclone male header socket. Attach the module in front of the header socket using foam tape. Alternate driving the left and right IR LEDs at 100Hz. Check for an IR sensor return and develop two signals, LEFT and RIGHT to indicate if the IR sensor return is from the left or right IR LED. The IR LEDs may need to be adjusted or shielded with some heat shrink tubing so that the floor does not reflect IR to the sensor. Use the LEFT and RIGHT signals to drive the decimal points on two LEDs to help adjust the sensor. It may be necessary to filter the IR returns using a counter with a return/no return threshold for reliable operation. Using a clock faster than 100Hz, for example 10kHz, only set LEFT or RIGHT if the return was present for several clock cycles.
9. Using IR sensor input, develop a design for the FPGA-bot that follows a person. The person must be within a foot or so of the FPGA-bot. When a left signal is present turn left, when a right signal is present turn right, and when both signals are present, move forward a few inches and stop. When all signals are lost, the FPGA-bot should rotate until an IR return is acquired.
10. Use motor speed control and a state machine with a timer to perform a small figure eight with the FPGA-bot.
11. Once the IR proximity sensor module from problem 8 is interfaced, design a state machine for the robot that moves forward and avoids obstacles. If it sees an obstacle to the left, turn right, and if there is an obstacle to the right, turn left. If both left and right obstacles are present, the robot should go backwards by reversing both motors.
12. With two FPGA-bots facing each other, develop a serial communications protocol using the IR LEDs and sensors. Assume the serial data is fixed in length and always starts with a known pattern at a fixed clock rate. The IR LEDs are pulsed at around 40kHz and the sensor has a 40kHz filter, so this will limit the bandwidth to a few kHz. Transmit the 8-bit value from the Cyclone DIP switches and display the value in the receiving FPGA-bot seven-segment LED displays. Display the raw IR sensor input in the decimal point LED to aid in debugging and alignment.
13. Interface the line-following module to the FPGA-bot, and design a state machine that follows a line. The line-following module has three sensor signals, left, center, and right. If the line drifts to the left, turn right, and if the line drifts to the right, turn left. Adjust turn constants so that the FPGA-bot moves along the line as fast as possible. If speed control was developed for the FPGA-bot as suggested in earlier problems, try using speed control for smaller less abrupt turns.
14. Using a standard IR remote control unit from a television or VCR and an IR sensor interfaced to the FPGA-bot, implement a remote control for the FPGA-bot. Different buttons on the remote control unit generate a different sequence of timing pulses. A digital oscilloscope or logic analyzer can be used to examine the timing pulses.
15. Interface the a magnetic or electronic compass module to the FPGA-bot, and design a state machine that performs the following operation:
    - Turn North.
    - Move forward 4 seconds.

- Turn East.
- Move forward 4 seconds.
- Turn Southwest.
- Move forward 6.6 seconds.
- Stop and repeat when the pushbutton is hit.

The mechanical compass has a small time delay due to the inertia of the magnetized rotor. Just like a real compass, it will swing back and forth for awhile before stopping. With care, the leads on the compass module can be plugged into a DIP socket with wire wrapped power supply and pull-up resistor connections on a small protoboard or make a printed circuit board for the compass with jumper wires to plug into the Cyclone female header socket. Make sure the compass module is mounted so that it is level and as far away from the motors magnets as possible.

16. Interface a Sonar-ranging module to the FPGA-bot and perform the following operation:
    - Scan the immediate area 360 degrees by rotating the robot
    - Locate the nearest object.
    - Move close to the object and stop.
17. Attach the Sonar transducer to an unmodified servo's output shaft. Use the new servo to scan the area and locate the closest object. To sweep the unmodified servo back and forth, a timing pulse that slowly increases from 1ms to 2ms and back to 1ms is required. Move close to the nearest object and stop.
18. Attach several IR ranging sensors to the FPGA-bot and use the sensor data to develop a wall following robot.
19. Interface additional sensors, switches, etc., to the FPGA-bot so that it can navigate a maze. If several robots are being developed, consider a contest such as best time through the maze or best time after learning the maze.
20. Use the μP 3 computer from Chapter 8 to implement a microcontroller to control the robot instead of a custom state machine. Write a μP 3 assembly language program to solve one of the previous problems. Interface a time-delay timer, the sensors, and the motor speed control unit to the μP 3 computer using I/O ports as suggested in problem 8.6. The additional machine instructions suggested in the exercises in Chapter 8 would also be useful.
21. Use a Nios processor to control the robot with C code using the UP3 Nios II reference design in Chapters 16 & 17.

22. Develop and hold a FPGA-bot design contest. Information on previous and current robotics contests can be found online at various web sites. Here are some ideas that have been used for other robot design contests:
    - Robot Maze Solving
    - Robot Dance Contest
    - Sumo Wrestling
    - Robot Soccer Teams
    - Robot Laser Tag
    - Fire Fighting Robots
    - Robots that collect objects
    - Robots that detect mines