



Car Assembly Manual



| | |
|-------------------------------|------|
| 1. Introduction | 3-8 |
| 2. Assemble Car Chassis | 9-30 |
| 3. Download Codes | |
| 4. Test Your Car | |
| 5. Be the Best | |

1. Introduction

1.1 What is The Freescale Cup (TFC)?

The Freescale Cup is a global competition where student teams build and program an autonomous model car, and race it around a track. The team who finishes the track in the shortest time without derailing wins. TFC is a great opportunities for students to learn about electrical engineering, mechanical engineering, software C-language programming, as well as improving communication and team work skills.

1.2 What Background Knowledge Do I Need?

If You are currently studying in a high school, then you don't need to have any additional background knowledge to race your car on a track at TFC. You will find yourself easier to understand some concepts in this guide if you are good at math, physics or computer science. And it will be a huge bonus if you have already been familiar with circuitry and C programming.

1.3 What Do We Encourage You to Learn?

By participating in TFC, you will have the chance to be exposed to the latest Microcontroller (MCU) technology. You will need to program the K40 microcontroller to enable the car run on the track. During this process, you will learn some basic concepts about Electrical Engineering (circuits), Computer Engineering (MCU structure) and C programming. It will be a great opportunity for you to explore the real semiconductor industrial world.

1.4 What Is A Kinetis K40 MCU?

If your autonomous model car can be described as a human, then the microcontroller unit (MCU) will be its brain. The kit provides you with the K40X256 microcontroller from the Freescale K40 MCU family.

A MCU is a short for microcontroller unit, or simply a microcontroller. It is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output (I/O) peripherals. Microcontrollers are used broadly in automatically controlled products and devices, such as automobile engine, remote control, office machines and etc.

The Kinetis K40 low-power microcontroller unit (MCU) is a 32-bit MCU which is built around the ARM Cortex-M4 core with USB 2.0 full-speed OTG and segment LCD display controllers. It features in 100MHz maximum operating frequency, 256 Kbytes of flash memory, 64Kbytes of static RAM operated under ultra-low-power.

This manual step-by-step guides you how to build an autonomous intelligent car using the Freescale Car kit and the Kinetis K40 MCU.

This manual consists of five chapters. The *Introduction* chapter provides you some brief information about the Freescale Cup intelligent car race as well as what you can expect from the Freescale Car Kit. The majority of this guide focuses on assembling the car chassis and installing the source code into the K40 MCU chip.

The last chapter, the *Be The Best* chapter is for advanced users. You will find high level of information about how to modify the source code thus to maximize the speed. In addition, advanced concepts of Electrical Engineering, Mechanical Control System and C programming can be found in this chapter. You don't necessarily need the last chapter to run the car normally. But if you want your car be the fastest, that chapter will give you the information you need.

2. Assemble Car Chassis

2.1 Assemble Car Chassis

This chapter covers the step-by-step guide about car chassis assembly from the components provided in the Freescale Car Kit.

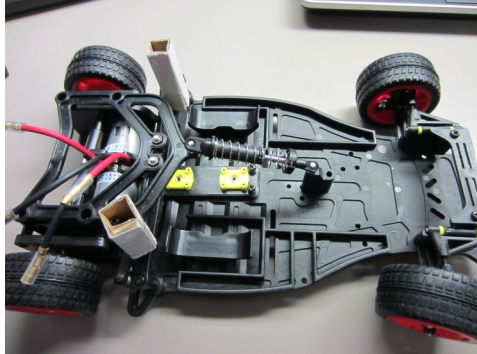
What is included in this chapter:

- Servo motor assembly
- Camera assembly
- K40 tower system building
- Detailed guide of wiring
- Hints for DIY
- Warning & Notice

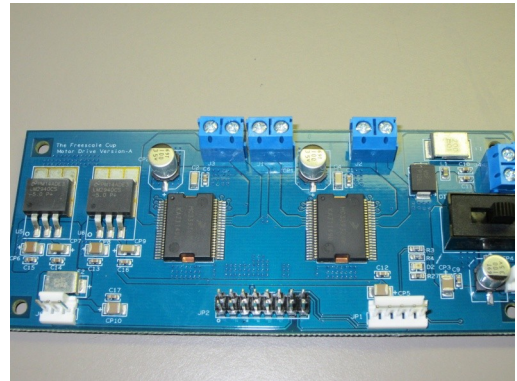
1. Set Up

Before your start, have the following components ready:

A. Model Car, Dual Motor X1



B. Motor Drive Board X1



C. Futaba Servo (S3010) X1



D. Accessory Kit Package X1



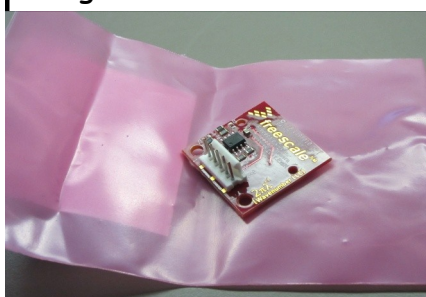
E. Wires/Camera Lens Package X1



F. Hardwar Kit Package X1



G. Light Sensor X1



*****Tools Recommended*****

- Screwdriver (large & small)
- Pliers
- Wire Stripper
- Modeling Knife
- Scissors

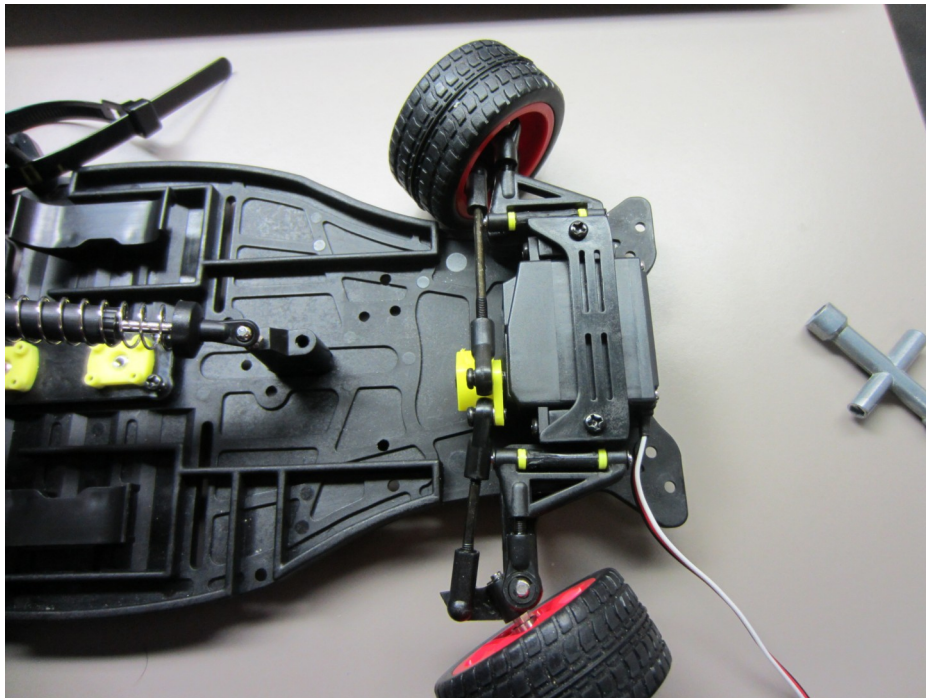
1. Attach the wheels

S1: Find the following in your kit package:

Nut X4, Grommet X4, Wrench X 1



S2: Attach the wheels, following the order:
wheel, grommet, nut

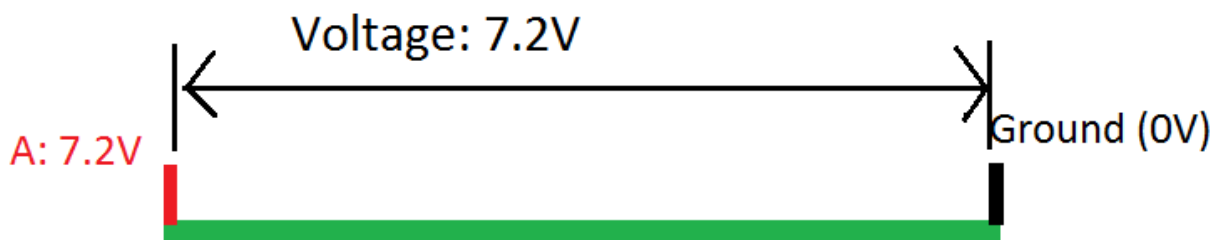


2. 1 Servo Motor

Servos are specialized DC motors that controls the steering of your model car. The servo motor is placed at the front of the Freescale Cup car, connecting the front two wheels with control arms.

The servo motor we use in the Freescale Cup is Futaba S3010, as shown in the picture. You can use a similar servo instead, as long as you can appropriately install the servo onto the car. Most modern servos have a three wire interface, as you can see coming out from your servo, labeled in colors of black (ground), red (V+), and white (control) separately.

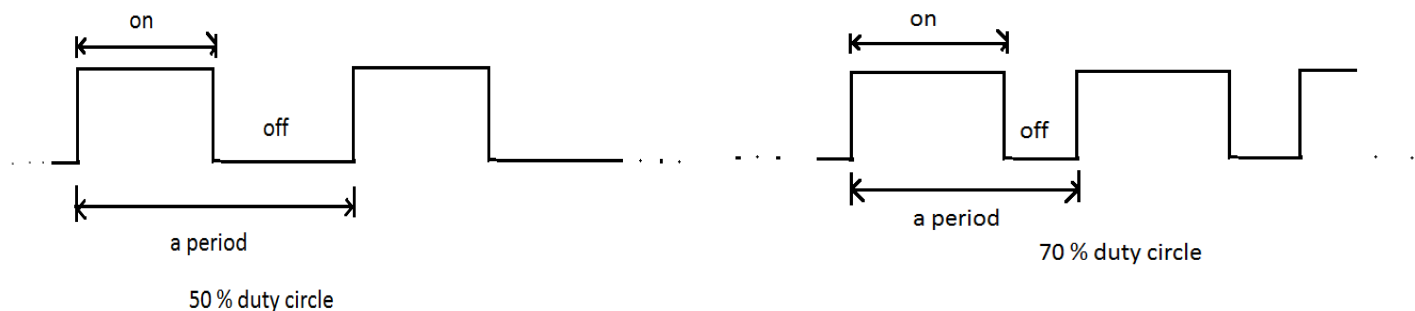
The black wire is for connecting to the **ground**. In electrical engineering, ground doesn't necessarily mean it is directly connected to the earth ground. It is a reference point (for example, 0 Voltage) where other voltages can be measured from. So for example, in the most cases, the ground voltage is set to be 0V. If the voltage at A is 7.2V, then the voltage difference will be:



$$7.2V - 0V = 7.2V$$

The red wire is for connecting to the **V+**, or a power supply source ranging from 4.8V ~ 6.0V depending on the speed of rotation you want. Here you don't need to worry about picking the voltage values. It will automatically set to 5.0V when you connect the servo to the board.

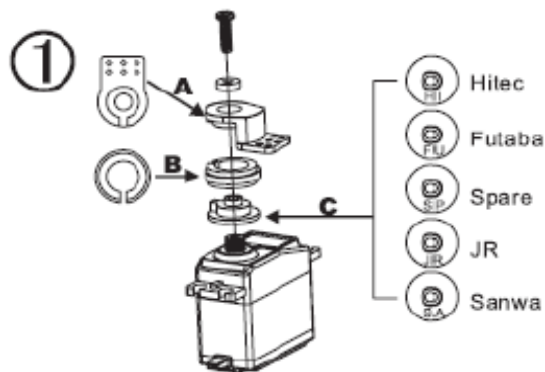
The white wire is for connecting the control signal, or pulse width modulation (PWM) signal. A PWM is a square wave with a set period. By changing the width of "on" versus the entire period, a digital ratio from 0 ~ 100% can be obtained. This ratio is called the **duty circle**, which is used to control the rotation of the servo. For example, 0 is full left, and 100% is full right.



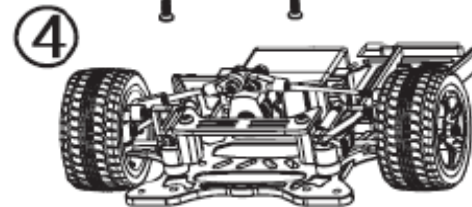
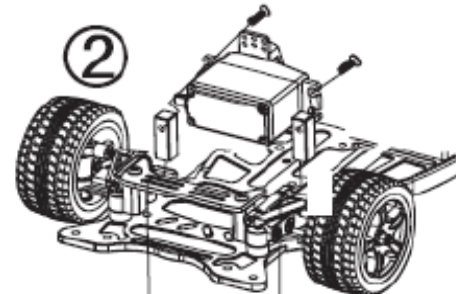
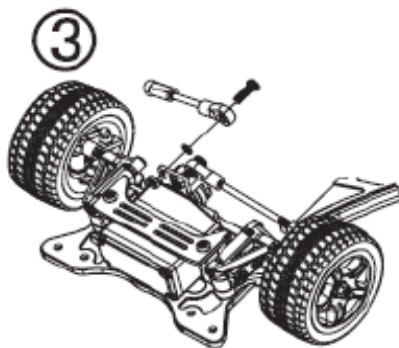
See the next page to see how to assemble and install the servo.

2.2 Assemble & Install the Servo

- In **step 1**, **DO NOT** force servo's steering position (the turning plate) using hands.
- There are two sizes of metal balls in the package. Pick the large ones.
- The control arms in **step 3** are quite tough to assemble. Use a pair of pliers to help to screw in the arm ends
- Make sure the servo is in the 0 degree position before securing the control arms in **Step 3**. Please refer to the next page (P13) to know more about how to electronically control the servo motor.
- After you set the servo motor to the 0 degree, you can first attach one arm, then hold the front wheel straight, then put on the other control arm. You will find this will help align the front two wheels, which is critical in the car's performance.

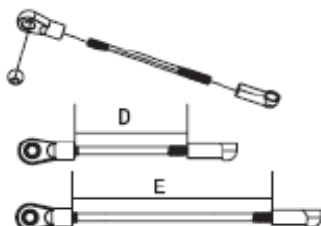


1. You can choose different plastic parts at A and B as desired.
2. Depending on the servo provided you must select the right plastic part in the C position.

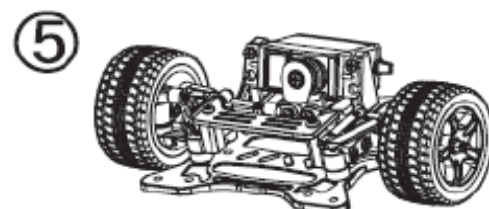
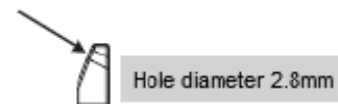


The servo can be placed to the front or backwards depending on the selected hole

- $\varnothing 5.8\text{mm}$ ball 2pcs
- ⬢ M3 nut 2pcs
- $\varnothing 3.1$ gasket 2pcs
- ⬢ M3X16 screw 2pcs
- ⬢ M3X8 sink screw 2pcs



By selecting pieces D or E you can adjust the left and right front wheel's angle.



2.3 Futaba S 3010 Specifications

Just for your reference. You don't need to know every part of this specs to make the servo work.



<http://www.gpdealera.com/cgi-bin/wgainf100p.pgm?I=FUTM0043>

This is the Futaba Standard Size Ball Bearing High Torque Servo.

This servo can produce high-current draw from your batteries.
If using NiMH or LiPo batteries, make sure they are capable of delivering approximately 2A for each servo.

FEATURES: Ideal for high-torque applications requiring a standard size servo
Universal connector fits Futaba, Hitec, JR, KO Propo, Airtronics Z, and Tower Hobbies. Does not fit old Airtronics A plug w/out adapter
Nylon gears
One bearing pre-mounted on output shaft.
One year warranty

INCLUDES: One Futaba standard size high torque servo with;
One 1.4" (35mm) diameter round servo wheel
One 1.5" (39mm) diameter 4 point servo wheel
One 1.25" (32mm) diameter 6 point servo wheel
Four 2mm x 11mm phillips screws
Four rubber grommets & Four metal eyelets

REQUIRES: Small phillips screwdriver to mount to surface

SPECS: Speed: 0.20 sec/60° @ 4.8V
0.16 sec/60° @ 6.0V
Torque: 72 oz-in (5.2 kg-cm) @ 4.8V and
90 oz-in (6.5 kg-cm) @ 6V
Dimensions: 1.6 x 0.8 x 1.5" (1-9/16 x 13/16 x 1-1/2")
(40 x 20 x 38mm)
Weight: 1.5oz (1-7/16oz) (41g)

3.1 Line Scanning Camera

The line scan camera module in the kit consists of a CMOS linear sensor array of 128 pixels and an adjustable lens. The camera has a resolution of 1X128 in one dimension, which is why it is called "line scan". The camera will be mounted above the car to look ahead the distance. By changing the camera's angle, the "look-ahead" distance can be changed to ensure more efficient steering. In simple, the farther the car look ahead, the earlier it can turn. Although, in reality, there are a lot of things to consider about, such as light condition, irrelevant environment, and etc. But they are not in the scope of this part of discussion.

Pins layout:

If you take out the sensor cable (refer to the next page), you will see five wires, labeled in color of black (ground), brown (Vdd), red (clock), orange (serial input) and yellow (analog out).

Ground: It connects to the GND pin. We have already talked about this in the previous chapter.

Vdd: It is the same as V+. It connects to a DC power source, which is 5V for this CMOS sensor.

Clock: All digital sensor needs a clock pulse signal to coordinate its actions. Basically, the clock signal oscillates between high and low in the form of 50% of duty cycle with a constant time period. The clock signal is vital for a digital sensor to function.



SI: A SI signal, or a serial input is a pulse signal that plays as shutter in a camera. This signal tells the camera to scan and expose the image for every couple milliseconds.



AOUT: Analog out is an analog signal (continuous signal) that is outputted and processed by the K40 microcontroller. The signal is continuous varying from 0V to Vdd, which is +5V in this case. An Analog to Digital Converter (ADC) in the microcontroller will convert this continuous signal into a series of discrete number (digitalization) so that the MCU can understand.

For example, an 8 bit ADC has 256 discrete levels ($2^8 = 256$). So 0V will correspond to a digital number of 0. And 5V will correspond to 255 (digital number starts from 0 not 1). The amount of increment of each discrete step will be:

$$5V / 256 = 0.01953125 \text{ V/bit}$$

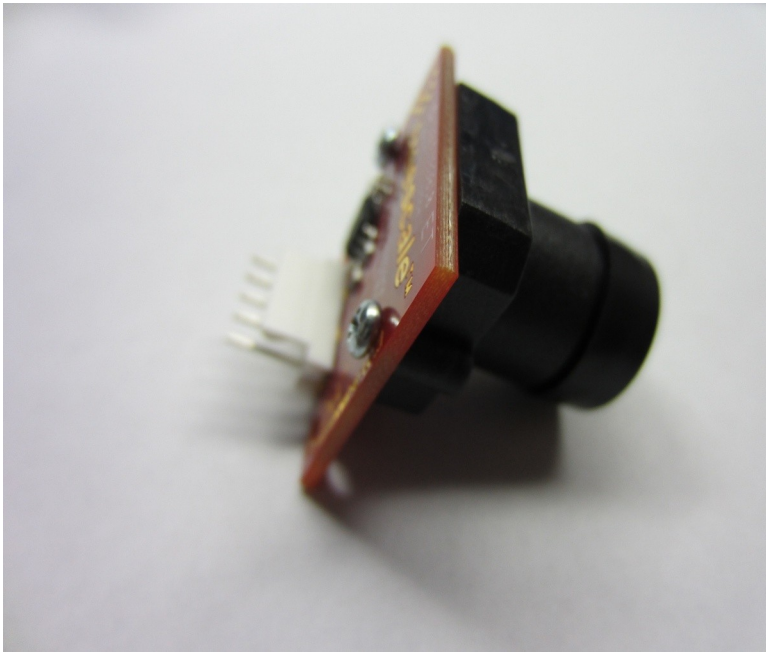
So bit 128 will be 2.5 V. ($0.01953125 \text{ V/bit} * 128 \text{ bit} = 2.5V$)

3.2 Assemble the camera

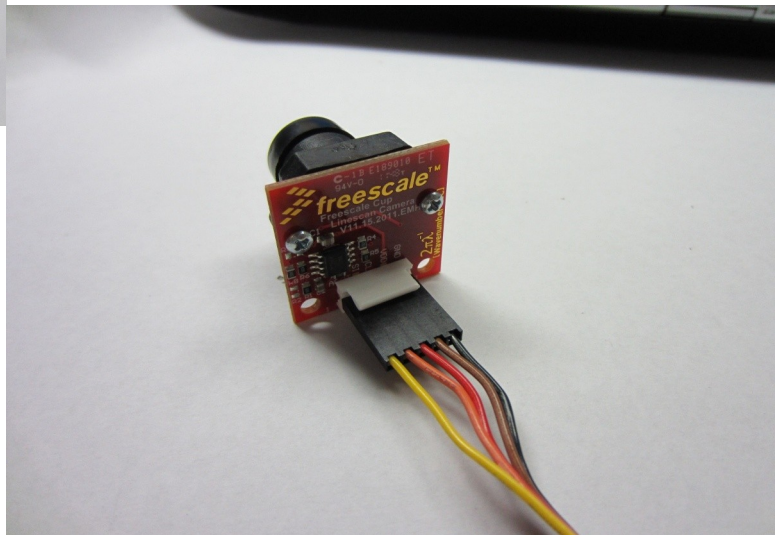
S1 : Take out the lens, light sensor and two screws from the package. Be careful, the sensor is **electrostatic sensitive**. You can touch a grounded metal to release your body's static electricity. A metal table leg should do the trick.



S2: Assemble the camera. The orientation doesn't matter. Watch out your screw driver not to damage small components on the sensor's back.



S3: Connect the module to the sensor cable. Refer to the labels at the back of the sensor, the black wire should be connected to **GND** and yellow to **AOUT**.



4.1 7.2V DC Battery

The 7.2V DC battery is the power source of your autonomous car. It supplies power for all the components on your car, from driving motors to the MCU. Different components require different supply voltages. So they are connected to a motor board, where on each corresponding port voltages are regulated. Then the battery connects to the motor board to power it up.

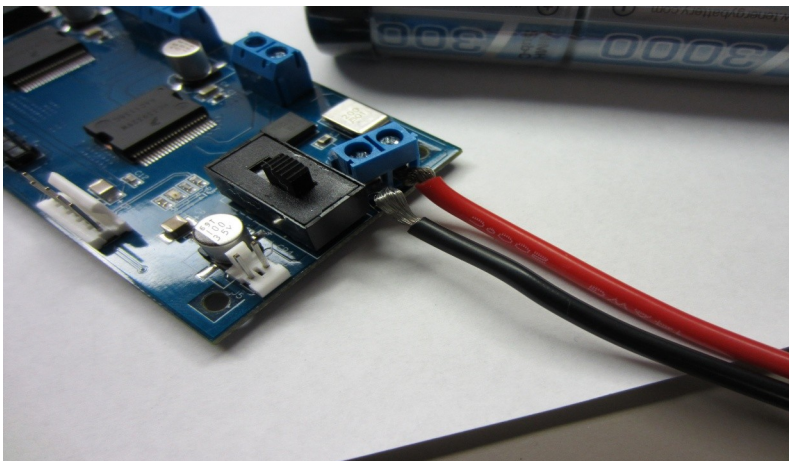
See the next page to see how to install the battery pack.

4.2 Install Your Batteries

S1: Connect your batteries



S2: Sharpen the wires' ends so that they can fit into the motor board(you don't necessarily need to plug them in now)

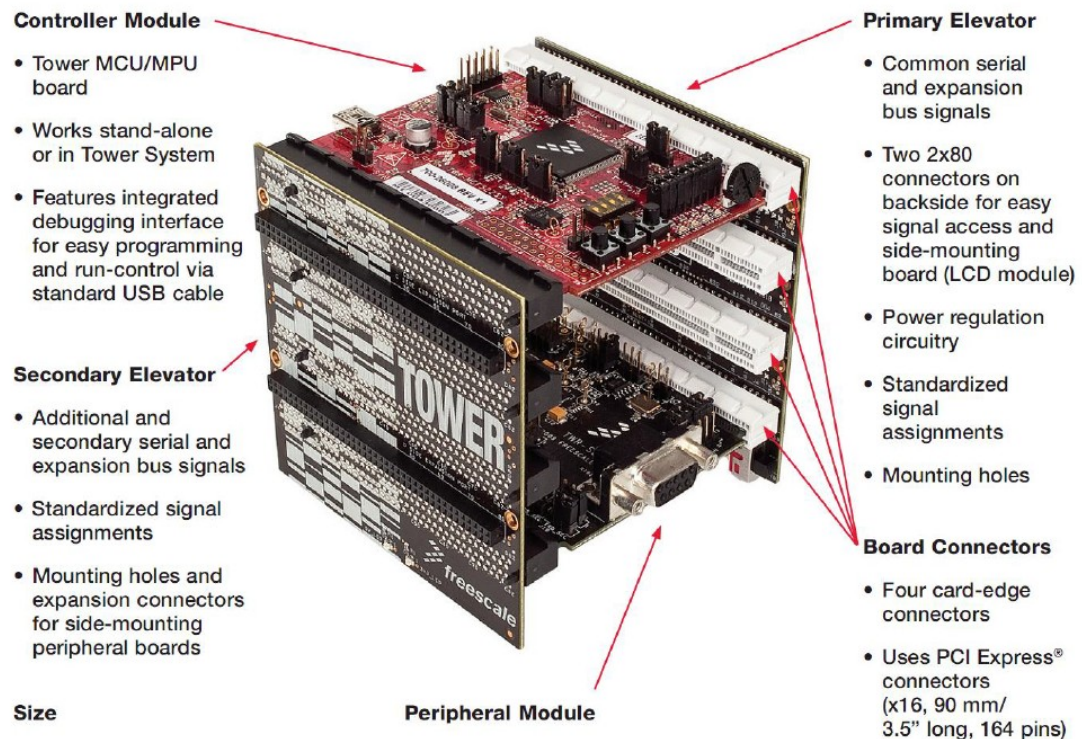


S3: Install your batteries. Don't forget to fix them.



5.1 K40 MCU Board and The Tower System

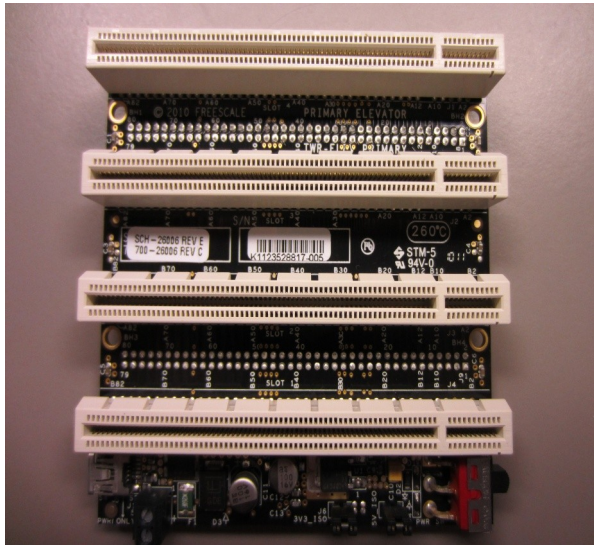
The Freescale Kinetis TWR-K40X256 tower system consists of the K40 tower controller module, the TWR-Elevator modules (primary and secondary elevator modules), TWR-Prototype module and TWR-SER module. The tower system is compatible to many peripherals from Kinetis K40 family. The tower system can be built up as a development platform to perform a variety of applications.



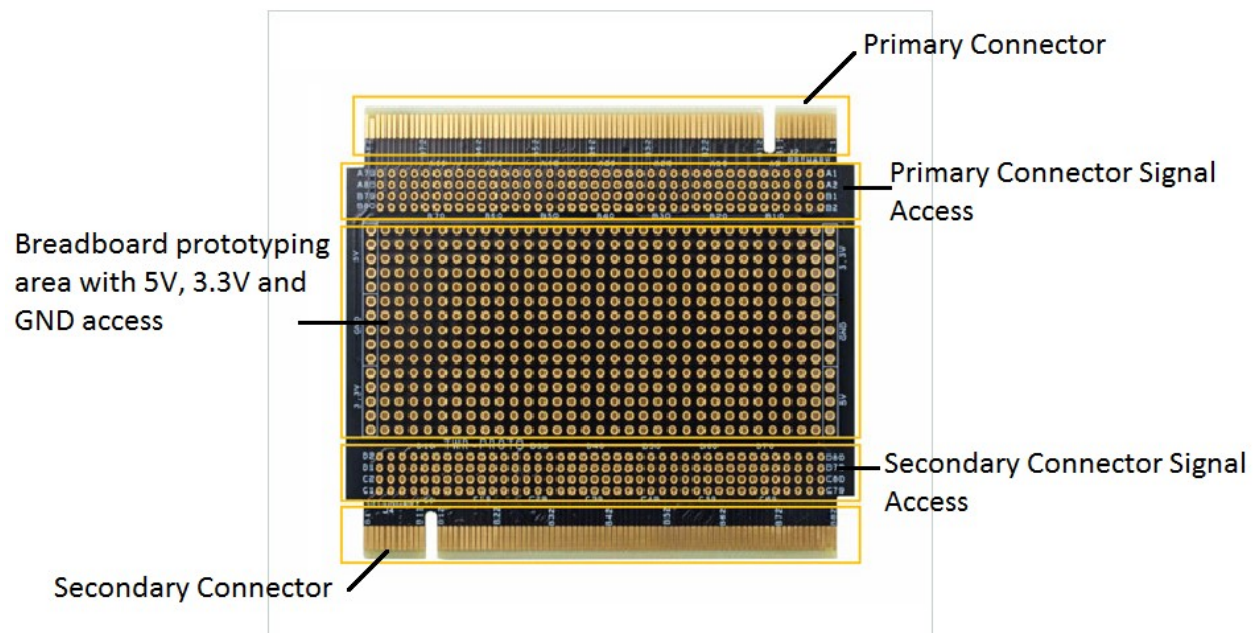
Kinetis TWR-K40X256 controller module: The K40 MCU board can function as a stand-alone platform as well as being included as the part of the K40 tower system. K40 board is the core of the whole tower system. Engineers read its reference manual and schematic to know which port and peripheral they can use for their applications. Then programs can be written and flashed into the chip. The process of programming a MCU can be very long and difficult. Fortunately, we have already composed the majority of the code for you. You will just need to find the code and download it into the chip, and maybe do a slight modification.

As you can tell, the MCU board you use looks different from the one showed in the picture. This MCU board, developed by Professor Eli Hughes from Penn State University, integrates with the motor drive board. This integration makes wiring much easier.

TWR-Elevator Modules: The TWR-ELEV Elevator modules, the Primary Elevator Module and the Secondary Elevator Module, are the basic building block of the K40 Tower System. Designed to connect microcontroller and peripheral modules, the Elevator modules provide the power regulation and structural integrity. In addition, the primary elevator board provides power inputs from a standard USB mini-B connector or a screw terminal. The TWR-ELEV Elevator modules feature four 164-pin card edge connectors.



TWR-PROTO Module: The TWR-PROTO Prototyping Module provides an easy way for designers to add custom circuitry to their Tower System designs. The TWR-PROTO module allows to be directly plugged into the Tower System. The perfboard area provides access to all of the signals as well as a 8.3 X 3.8cm prototyping area.



5.2 Assemble K40 Tower

Now you have some basic ideas about what a Freescale Tower System is. It is time for you to assemble your own Tower System. The Tower System can be compatible with many Freescale peripheral modules. But for the Freescale Cup Intelligent Car Race, you will only need the followings (all included in your kit):

Kinetis K40X256Z MCU Controller Module X 1

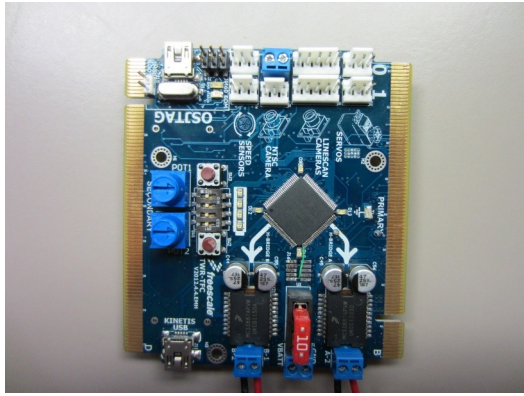
TWR-ELEV Primary Elevator Module X1

TWR-ELEV Secondary Elevator Module X1

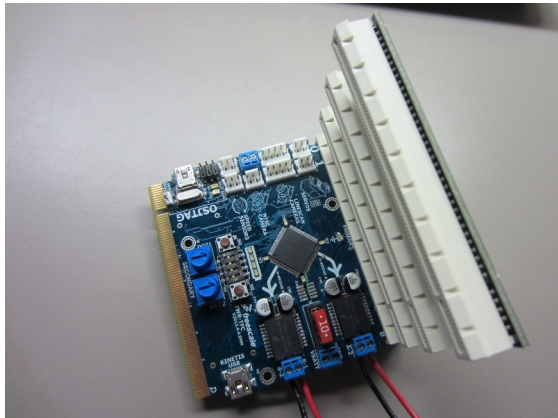
TWR-PROTO Module X1

5.2 Assemble K40 Tower

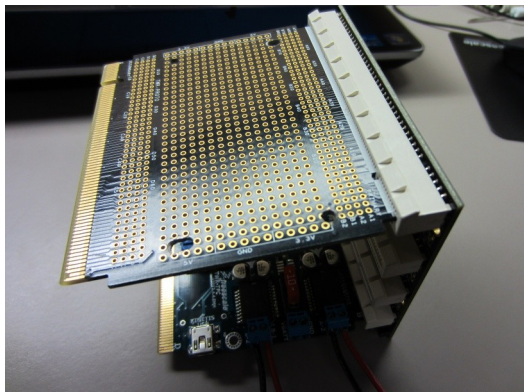
S1: Prepare the K40 MCU Controller Module as the base.



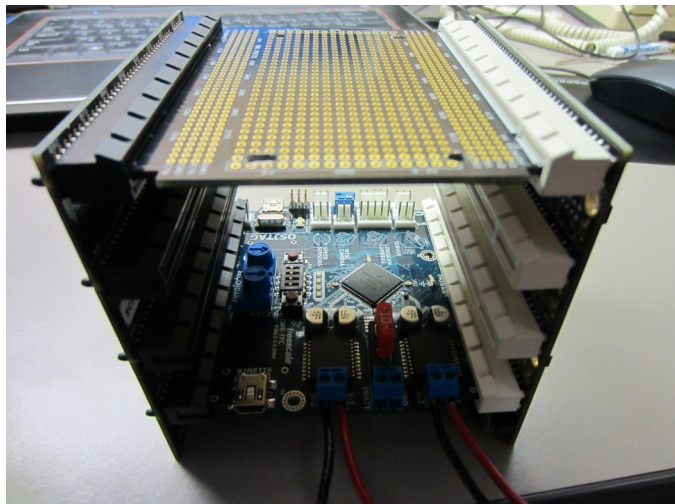
S2: Connect the MCU to the Primary elevator, labeled by its white edge.



S3: Plug in the Proto Module with its white card edge.



S4: Connect the Secondary Elevator Module to the system, labeled by its black edge.

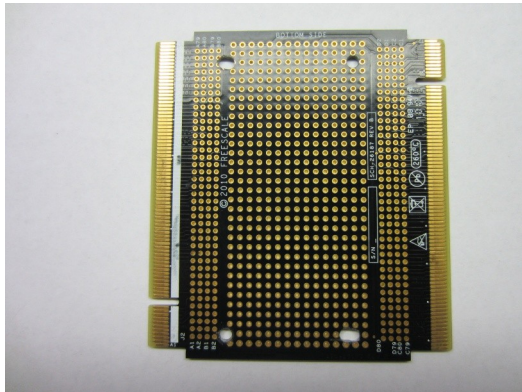


Now you have built your K40 Tower. You will have to find a way to mount the tower on your car chassis. The Freescale Cup Car Kit doesn't provide board mounting gears to mount the tower. So you will need to DIY your own mounting.

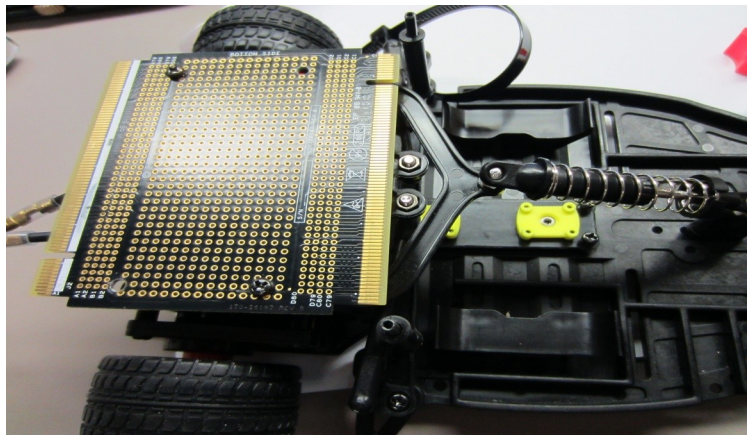
One way of doing this is attaching the TWR-PROTO board on the car chassis. By doing this, you will need to drill 4 holes on corners of the PROTO board. There are a lot of bus on the PROTO board. So you don't need to worry about damaging the whole board. There is no standard for the mounting design. Just let your imagination fly! You probably will make something that are both useful and nice-looking if you are willing to brain storm.

Here is an example for mounting your K40 TWR board onto the car chassis.

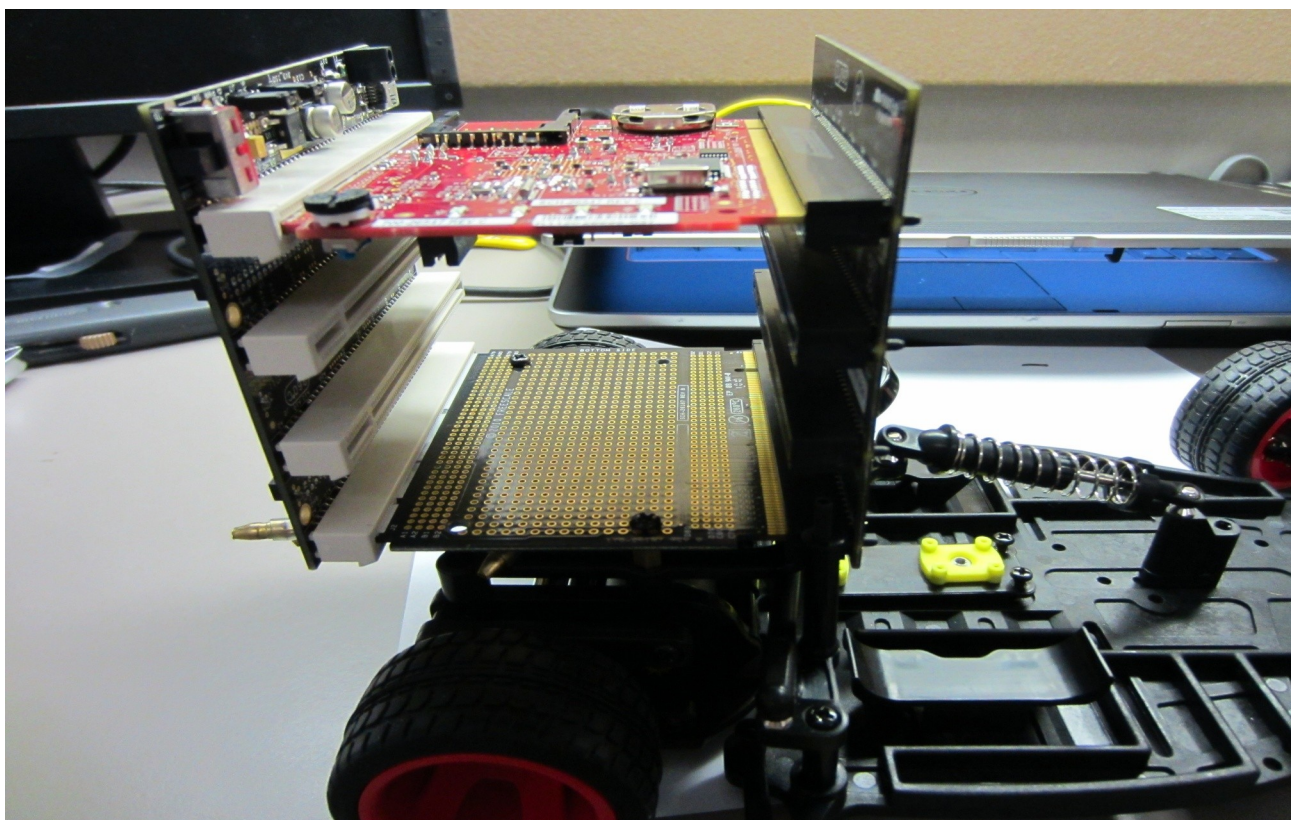
S1: Drill 4 holes on the prototyping board.



S2: Place the prototyping board on the car chassis, you may need 4 pillars for lifting it up a little bit.



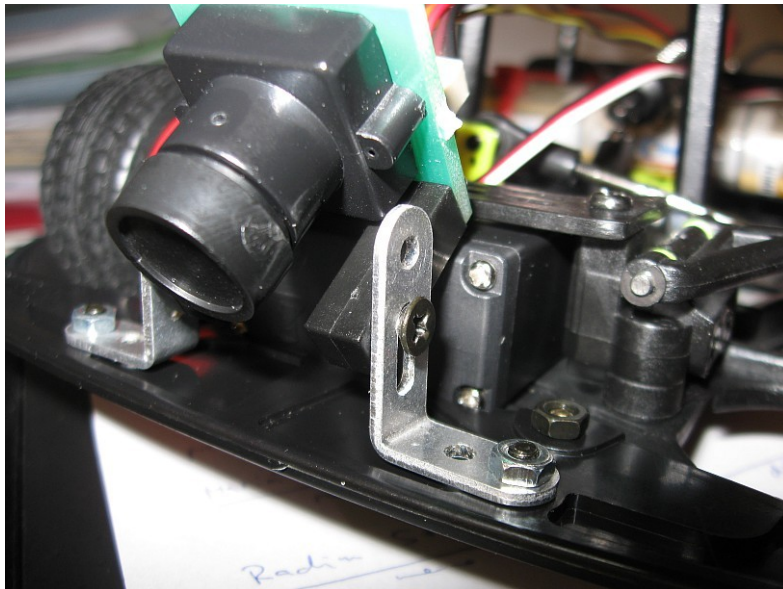
S3: Build your tower. Notice that the primary elevator module should be attached to the primary connectors, which are indicated by the white edges on the PROTO board and the Kinetis K40.



6.1 DIY your camera mounting

The line scanning camera line scans the track and send data back to the microcontroller. The kit doesn't provide any mounting components for mounting the camera. So you will have to DIY your own way to appropriately mount the camera onto the car chassis. You probably won't want to fix the angle and height of your camera because you may need to test different settings for optimizing the image captured.

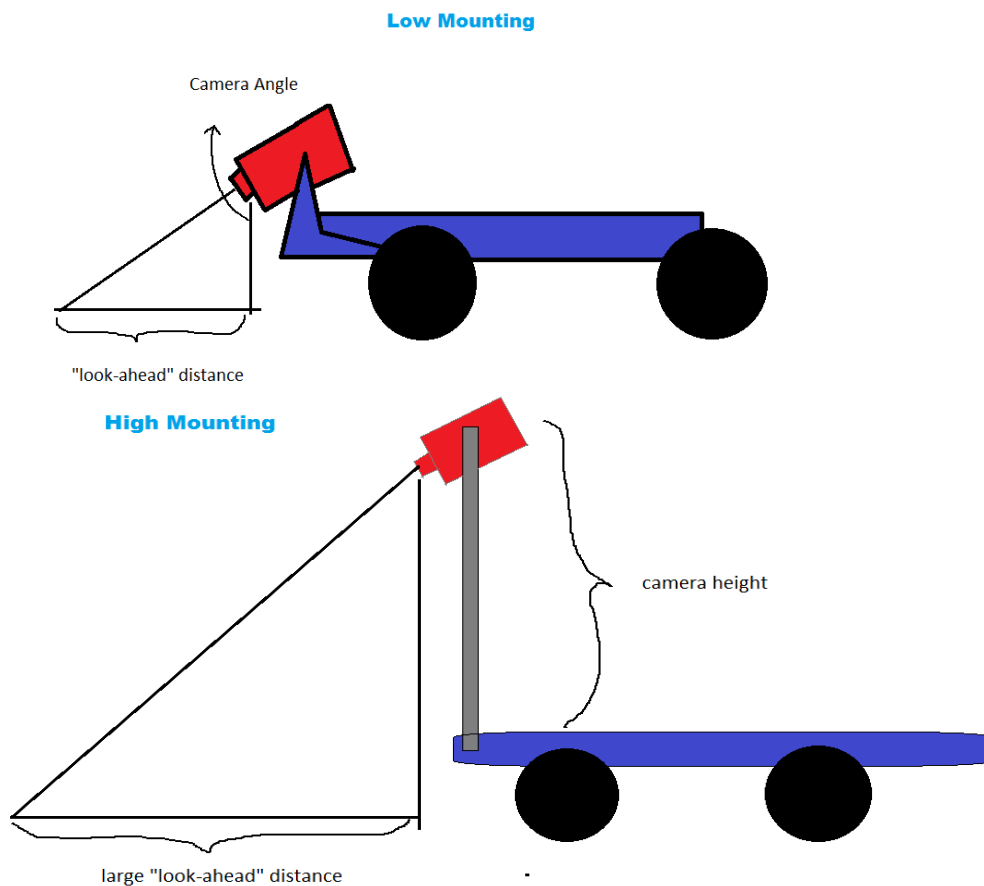
Here is one example showed in the picture below. Two metal L-shaped aluminum pieces and two black plastic distance posts (already in the kit) were used in this design. You can also use a couple of plastic pieces and a long metal rod to mount the camera high above the car chassis. No matter how do you design your camera mounting, make sure that the camera gets hold steady when your car runs. Because shaking of the camera is very bad for image data processing.



6.2 Camera Mounting Strategy

Unlike the Tower mounting, the way that you mount your camera can probably affect the performance of your car. The example showed in the previous page allows the camera look only very short distance ahead, which doesn't give much time for the servo to steer. For example, in reality, if you are driving a car in a big fog (your eye sight is greatly limited), then you will have to drive very slowly so that you will not hit other cars or people, or run off the road. For the same reason, the shorter the look-ahead distance, the slower the speed of the car must run to ensure it will not go off track when it turns.

A better design will be mount the camera at the height about 4~6 inches above the car chassis. Having a hinge will make it better to ensure you can adjust the camera's angle large or small. But keep in mind that, it is not that the look-ahead distance is the longer the better. Greater sight will also add in more irrelevant information to the camera, thus add more irrelevant data for the MCU to process. Our programmer had a "filter" algorithm in the program to filter out irrelevant data (shadows, side tracks, light distortion, etc). But it will be better if you don't set the look-ahead distance too large (too high mounting, or to large camera angles).



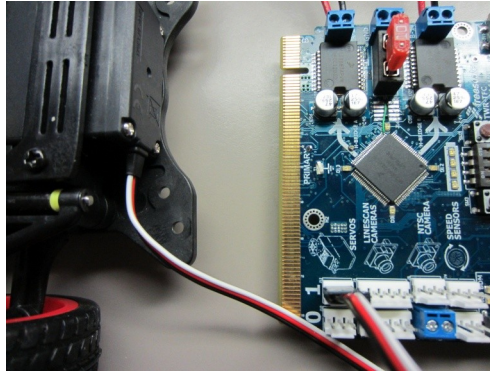
the MCU Board

In this chapter, you will know how to connect all components to the MCU board. This MCU board is a combination of K40 MCU Controller Module with motor drive board. Engineers usually need to read the schematic of the motor drive board to know which pin on the MCU board has to connect to which pin on the motor drive board. This process can be quite time consuming. Fortunately, the new board was designed specifically for The Freescale Cup Car Race. All you need to do is just plugging each components to the correct position on the board.

the MCU Board

S1: Connect the camera to the board. Use Pin J8 (the row "1") not J9.

S2: Connect the servo motor to the board. Use J12. Be careful with the orientation, look for the "GND" pin.

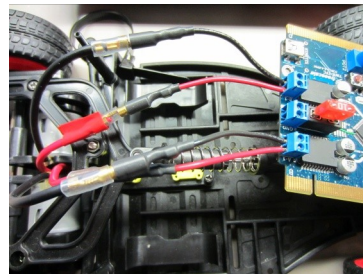
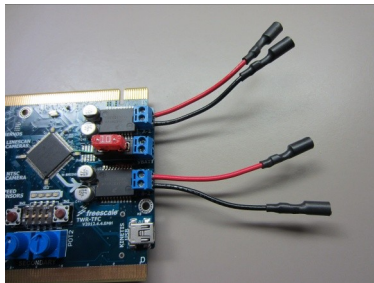


S3: Connect the motor to the board. Use J3 and J4.

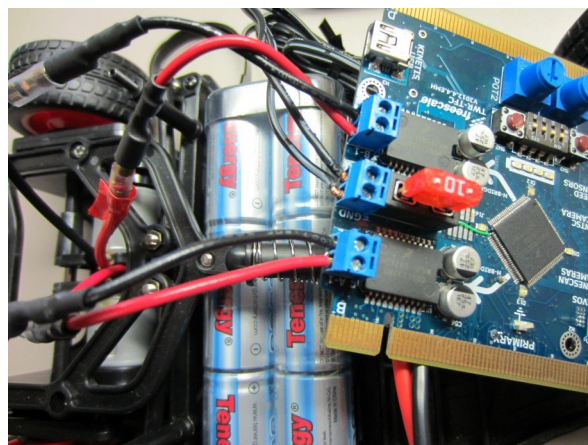
I. Put in four connecting wires.

II. Connect them with the driving motor.

Make sure you match the wires' color.



S4: Connect the battery to the board. Again, be careful with the orientation, the ground line **MUST** go to "GND" port.



1. The motor cables are not soldered very tightly. You can easily pull them off, so be careful when you are connecting the motor cables.
2. Servo motor assembly: It will be much easier if you attach one control arm first, then have one person hold the wheel straight and attach the other one.
3. Use electrical tape to wrap any exposed wire. Exposed wires can potentially cause the board short-circuited and burn the chip!
4. **Never** power the microcontroller from different power sources! Switch off the MCU board before you connect your it to your PC.
5. Be careful with the orientation when you plug in wires. It is a good idea if you can verify the right pin with a digital multimeter (DMM).

3. Test Your Car

3.1 Testing and Debugging

A typical engineering process follow the sequence of designing, prototyping, testing and debugging. It is a repeated sequence. The problems an engineer finds during his testing may drive him back the first stage, which is designing. That is why many big engineering projects, such as aerospace projects and military projects can take many years. Sometimes some of them never got completed.

To make it more appropriate for high school students, the Freescale has already done the designing and prototyping stage for you. Our engineers wrote the code based on the design specifications that were given to them.

It is rare for an engineering design that success the first time you test it. So you'd better be prepared for the failure when you first test your car. One very important thing for you to learn, is how to find out what the problem is causing the failure. There can be many facts that can cause the failure.

For example, if your car goes off the track when it turns. The "bugs" can be from the distortion of environmental light, some loose attachment from the wheels, angle of your camera, or even the designing aspect of your code algorithm (that will be very nasty). In this case, write a list of every facts you can think of that can cause it. Try to fix them one by one. Tools will be very helpful when you want to verify that you have really fixed a potential problem. For example, if you are afraid of that you made a wrong setting in your program that causes zero voltage output on pin J16 on the board. Use a multi-meter to measure the voltage between J16 and the ground to see if it does read zero voltage.

As I just said, testing and debugging is repeated process. Be patient, if you don't fix the problem when you debug it the first time. It is very common that debugging takes very long time. Actually, the majority time of any engineering design is spent on testing and debugging.

