Jeremy Cooley

GGC – ITEC 4400 – Internet of Things – Lawson

4/19/2019

Adeept PiCar-B – Project Evaluation

**1. Purpose of the Device (as an IoT device)**  
 The Adeept PiCar-B is an educational, smart car robot kit that aims to familiarize customers with sensors, actuators, and data transmission as a facet of the Internet of Things. As a company, Adeept is a “technical service team of open source software and hardware […] Dedicated to applying the Internet and the latest industrial technology in open source area, [and striving] to provide the best hardware support and software service for general makers and electronic enthusiasts around the world.” The assembly of the robot was complex and took a lot of time to complete, but the Graphical User Interface of the software included with the PiCar-B Although the PiCar-B comes with its own “hat” for connecting the motor and sensors to the Raspberry Pi’s GPIOs, in the future, I would like to research how to attach more or different sensors to the Pi & PiCar to alter the functionality of the robot. Currently, the software that comes with the robot kit is similar to that in self-driving cars: Distance–Sensing software, Distance–Maintaining software, Line–Following software, Voice/Speech Recognition and Audio–Processing softwares, and even Object–Tracking software!

**2. Summary of how the device was assembled (all parts & software)**

**The full manual we followed can be found at:** [**https://www.adrive.com/public/9hCcHR/Adeept\_PiCar-B-V1.0.zip**](https://www.adrive.com/public/9hCcHR/Adeept_PiCar-B-V1.0.zip)

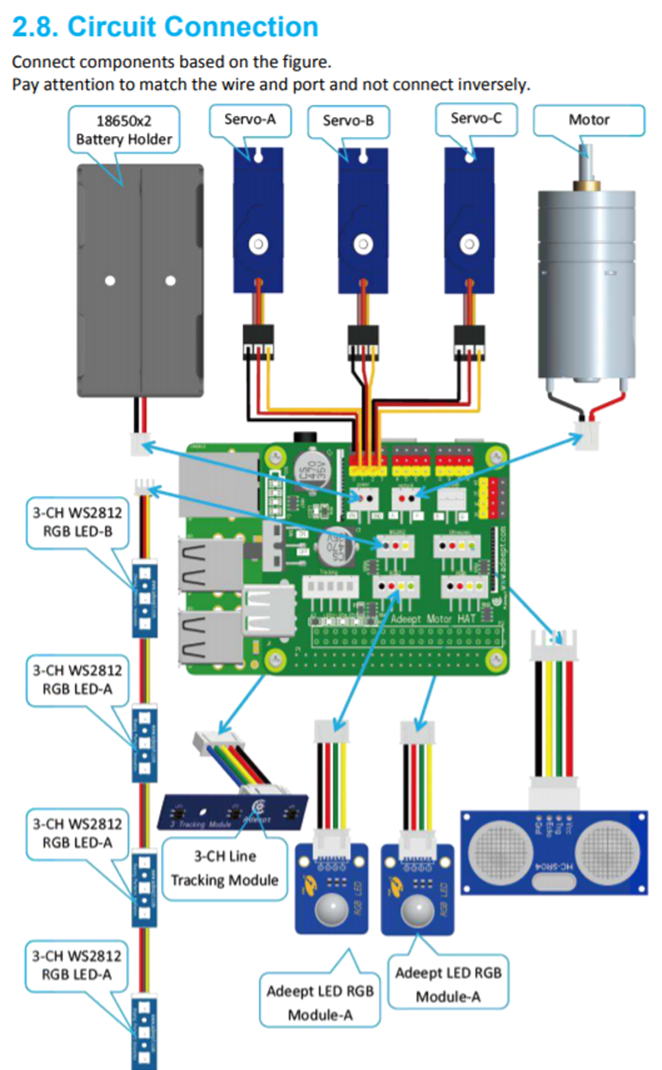
**Features:**

* STEM Educational Robot - An complete AI(Artificial Intelligence) robot kit based on the Raspberry Pi(Compatible with RPi 3B/3B+/2B/2B+, Raspberry Pi is NOT included).
* Speech Recognition - PiCar-B can be controlled by voice; Object Recognition and Tracking - based on openCV; Automatic Obstacle Avoidance - based on ultrasonic sensor; Line Tracking - based on infrared reflection; C/S Architecture - can be remotely controlled by APP on PC; WS2812 RGB LEDs - can change a variety of colors, full of technology; Real-time Video Transmission.
* Easy to Assemble and Coding - A 81-page PDF manual with illustrations is considerately prepared for you, which teaches you to assemble your Raspberry Pi robot step by step; Easy-to-understand Python code is provided, with simple and practical GUI program (compatible with Windows and Linux operating systems).
* Strong Technical Support - Official forum, Blog, YouTube video and instant feedback with E-mails.
* **Powered by 2x 18650 batteries (NOT included).** You need to prepare your own batteries.

**Package List:** 1 Set Acrylic Plates  
 1x Adeept Motor HAT V2.0  
 1x Raspberry Pi Camera(with Cable)  
 1x USB Microphone  
 1x Ultrasonic Sensor Module  
 2x Adeept RGB LED Module  
 4x Adeept WS2812 RGB LED Module  
 1x Adeept 3CH Line Tracking Module  
 3x Servo  
 1x Gear Motor  
 4x Wheels  
 1x Battery Holder  
 1x Cross Socket Wrench  
 2x Cross Screwdriver(Small and Large)  
 1x Winding Pipe  
 10x Bearing(6\*F624ZZ + 4\*F687ZZ)  
 2x Umbrella Gear Set

**PiCar-B Assembly:**

* Identified, Labeled, and Sorted the nuts, screws, spacers, bearings, sensors, servos, and acrylic frame pieces. ( which each had paper covers that had to be removed; a tedious extra step! )
* Assembled the acrylic frame pieces with screws and nuts.
* Attached battery holder and RGB LED under-lights.
* Attached/Connected the PiCar’s motor, umbrella gear, rear axle, and rear wheels.   
  ( We struggled a bit getting the bearings to sit within the acrylic frame! )
* Calibrated the front wheel servos.
* Built the the front frame and and mounting the servos that turn the front wheels.
* Connected the front and back ends and adding in the slick RBG LED “Headlights” that indicate the PiCar-B’s Status.
* Assembled the “Transformer head” with the Ultrasonic Ranging Module, Camera, and servos with rocker arms that turn the head up and down and the “neck” left and right. ( I learned that a perpendicular assembly of servos like these are called gimbals—they enable or resist motion in multiple planes, like gyroscopes. Electro-Mechanical Engineer Dad For the Win! )
* Mounting the Head to the front of the acrylic frame.
* Mounted the Sensor Hat to the Raspberry Pi with spacers and screws.
* Mounted the Raspberry Pi / Sensor Hat Combo to the acrylic frame we’ve made!
* Plugged in the USB Voice Module to the Raspberry Pi.
* Connected the sensors, servos, LEDs, and Put the Batteries in.   
  ( See Diagram on next Page )



**Raspberry Software Installation:**

* We set up this new Raspberry Pi with Raspbian on a new SD card
* Acquired the Raspberry Pi’s IP Address
* Then used PuTTy to SSH into it.
* Downloaded and Ran Adeept’s PiCar-B **Setup** program   
   git clone <https://github.com/adeept/Adeept_PiCar-B.git>  
   sudo pythonAdeept\_PiCar-B/server/setup.py
* Because not all of the software was installed correctly the first time, I had to go back and do it manually:
  + Updated the System  
     sudo apt-get update  
     sudo apt-get upgrade
  + Enabled the **I2C** and **Camera Interfaces** through the settings menu on the Pi.  
    *“The Adeept Motor HAT V1.0 communicates with the Raspberry Pi via the I2C port but the I2C port is disabled by default. You need to enable it.”*  
     sudo raspi-config  
    Go To: 5 Interfacing Options->P5 I2C->Yes->OK  
    And then: 5 Interfacing Options ->P1 Camera ->Yes ->OK ->Finish ->Yes
  + Reboot, then check that the modules have been enabled:  
     reboot  
     lsmod | grep i2c
  + Installed the **I2C Tools**:  
     sudo apt-get install i2c-tools
  + Installed the Python program for the **RGB LEDs**:  
     sudo pip3 install adafruit-pca9685
  + Installed **Pip**, **Setuptools**, and **Wheel** for ease of use with Python:  
     sudo pip3 pip setuptools wheel
  + Installed **portaudio-19-dev**  and **python3-all-dev**, pre-req’s for PyAudio:  
     sudo apt-get install portaudio19-dev  
     sudo apt-get install python3-all-dev
  + Installed **PyAudio**, which *“can input sound from microphone”*  because, *“we use microphone input, otherwise it will raise an AttributeError when the programs [sic] attempting to instantiate a microphone object.”*:  
     sudo apt-get install python3-pyaudio  
     sudo pip3 install pyaudio
  + Installed **Flac *“****to convert the sound from microphone recordings into .flac files for PocketSphinx speech recognition:”*  
     sudo apt-get install flac
  + Installed the latest versions of **SphinxBase** and **PocketSphinx**:  
     sudo wget [https://sourceforge.net/  
    projects/cmusphinx/files/sphinxbase/5prealpha/sphinxbase-5pre  
    alpha.tar.gz/download](https://sourceforge.net/projects/cmusphinx/files/sphinxbase/5prealpha/sphinxbase-5preal%20pha.tar.gz/download%20-O%20sphinxbase.tar.gz) -O sphinxbase.tar.gz  
     sudo wget [https://sourceforge.net/  
    projects/cmusphinx/files/pocketsphinx/5prealpha/pocketsphinx-5pre  
    alpha.tar.gz/download](https://sourceforge.net/projects/cmusphinx/files/pocketsphinx/5prealpha/pocketsphinx-5p%20realpha.tar.gz/download%20-O%20pocketsphinx.tar.gz) -O pocketsphinx.tar.gz
  + Extracted the Tarball files into separate Directories  
     sudo tar -xzvf sphinxbase.tar.gz  
     sudo tar -xzvf pocketsphinx.tar.gz
  + Installed **Bison**, which *“converts scripts into a C language program that the computer can execute”* ( <https://whatis.techtarget.com/definition/Bison> )  
     sudo apt-get install bison
  + Installed **ALSA**, which *“provides API’s for sound card device drivers;”*   
    ( <https://en.wikipedia.org/wiki/Advanced_Linux_Sound_Architecture> )  
     sudo apt-get install libasound2-dev
  + & Installed **Swig**, which connects *“computer programs or libraries written in C or C++ with other scripting languages, in our case, Python.”*  
    ( <https://en.wikipedia.org/wiki/SWIG> )  
     sudo apt-get install swig
  + Then: (I have no idea why this is here b/c the manual doesn’t explain…)  
     sudo apt-get install python3 python3-dev python3-pip build-essential libpulse-dev
  + Compiled **Sphinxbase**: <https://github.com/cmusphinx/sphinxbase>   
     cd sphinxbase-5prealpha  
     ./configure –enable-fixed  
     make  
     sudo make install  
     sudo pip3 install pocketsphinx
  + Compiled **Pocketsphinx**: <https://github.com/bambocher/pocketsphinx-python>  
     cd ../pocketsphinx-5prealpha  
     ./configure  
     make  
     sudo make install
  + Installed **SpeechRecognition**, which *“is a library for performing speech recognition, with support for several engines and APIs, online and offline”*  
    ( <https://github.com/Uberi/speech_recognition> )  
     sudo pip3 install SpeechRecognition
  + Installed **Libopencv-dev**, which is a meta package that *“has dependencies to many packages that do contain the necessary libraries and header files that OpenCV require. [sic] ”*  
     sudo apt-get install libopencv-dev
  + Installed **imutils**, which *“contains a series of convenient functions which make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV.”*   
    ( <https://github.com/jrosebr1/imutils> )  
     sudo pip3 install imutils
  + Installed **OpenCV**, which is/are *“libraries of Python bindings designed to solve computer vision problems”* <https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_setup/py_intro/py_intro.html>  
     sudo apt-get install python-opencv  
     sudo pip3 install opencv-python
  + Installed **libatlas-base-dev**, **libjasper-dev, libqtgui4**, **python3-pyqt5**, and **libqt4-test**, which are all *“Python packages for FPV functions and sending jpeg stream to PC”* and include *“the static libraries and symbolic links needed for program development.”*   
     sudo apt-get install libatlas-base-dev libjasper-dev libqtgui4   
     python3-pyqt5 libqt4-test
  + Installed **Zmp** (**ZeroMQ**), which “Carries messages across inproc, IPC, TCP, TIPC, multicast.” ( <http://zeromq.org/> )  
     sudo pip3 install zmq
  + Installed **pybase64**, which “provides a fast base64 implementation for base64 encoding/decoding” ( <https://github.com/mayeut/pybase64> )  
     sudo pip3 install pybase64
  + Installed the Raspberry Pi **library** for controlling the **WS281X LEDs**  
     sudo pip3 install rpi\_ws281x
  + Downloaded the **Program** for the **PiCar-B** *(finally…)*  
     git clone <https://github.com/adeept/Adeept_PiCar-B.git>
  + Set up the Raspberry Pi’s **AP-Hotspot**  
     *“Configure the Raspberry Pi as the Wi-Fi hotspot mode to build up a direct communication between the PC and the car. If the RPi Car starts and there is no Wi-Fi to connect with, the RPi Car will set a AP-Hotspod [sic] itself. You can connect it with your PC:  
    Search Wi-Fi SSID name:* AdeeptCar *Password:* 12345678 *”*  (Adeept PiCar-B Manual, page 64)  
     git clone https://github.com/oblique/create\_ap  
     cd create\_ap  
     sudo make install
  + Installed the AP-Hotspot’s dependent libraries:  
     sudo apt-get install util-linux procps hostapd iproute2 iw  
     haveged dnsmasq
  + **Added** the RPi **Car Program** to **Auto-Start** & Create a car.desktop to add the RPi Car program to auto-run  
     sudo nano /home/pi/.config/autostart/car.desktop  
     To the file, add:  
     [Desktop Entry]  
     Name=Car  
     Comment=Car  
     Exec=sudo python3 /home/Adeept\_PiCar-B/server/server.py  
     Icon=false  
     Terminal=false  
     MutipleArgs=false  
     Type=Application  
     Catagories=Application;Development;  
     StartupNotify=true  
    Then press Ctrl + X to exit editing the file, and then Y for yes to save, and  
    then Enter to confirm.
  + Copied the **set.txt** to **/home/pi/***“so that the program could find it and load settings. If you set a wrong setting and the car won’t work well, you can simply copy it again to replace the old one, and the car will go back to normal.”* (pg 66)  
     sudo cp -f /home/Adeept\_PiCar-B/server/set.txt /home/pi/set.txt

**PC Software Installation**

* On our laptops, installed **Python 3.7**, the latest version: python.org >>> Downloads >>> Download Python 3.7.0 >>> Click OK. In the Command Line, type:  
   python -m pip install –upgrade pip setuptools wheel
* Installed **SpeechRecognition**   
   pip install SpeechRecognition
* Installed **PyAudio** using pip:  
   pip3 install PyAudio-0.2.11-cp37-cp37m-win\_win32.whl
* Downloaded & Installed **Swig** from:  
   <http://www.swig.org/download.html>
* Set the **Path Variable** for Swig.
* Downloaded **PocketSphinx.whl** from:  
   <https://www.lfd.uci.edu/~gohlke/pythonlibs/>  
   (why I’m not installing Swig here too, who knows)
* Installed the PocketSphinx.whl:  
   pip3 install pocketsphinx-0.1.15-cp37-cp37m-win\_amd64.whl
* Install **NumPy**, which is “a general-purpose array-processing package designed to efficiently manipulate large multi-dimensional arrays of arbitrary records without sacrificing too much speed for small multi-dimensional arrays”  
   pip3 install numpy
* Download **OpenCV\_Python.whl** from:  
   <https://www.lfd.uci.edu/~gohlke/pythonlibs/#opencv>
* Installed OpenCV\_Python with:  
   pip3 install opencv\_python-3.4.3-cp37-cp37m-win\_amd64.whl
* Installed **Zmp** (**ZeroMQ**) ( see above or @ <http://zeromq.org/> )  
   pip3 install zmq
* Installed **pybase64** ( see above or @ <https://github.com/mayeut/pybase64> )  
   pip3 install pybase64

**3. Summary of your testing methods and any surprises you found**

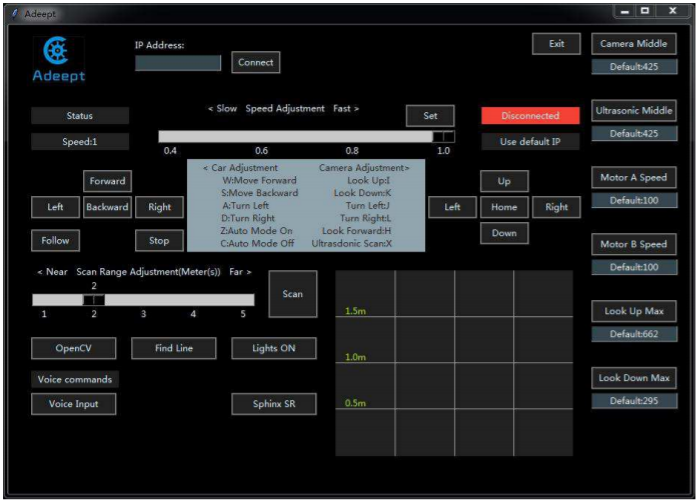
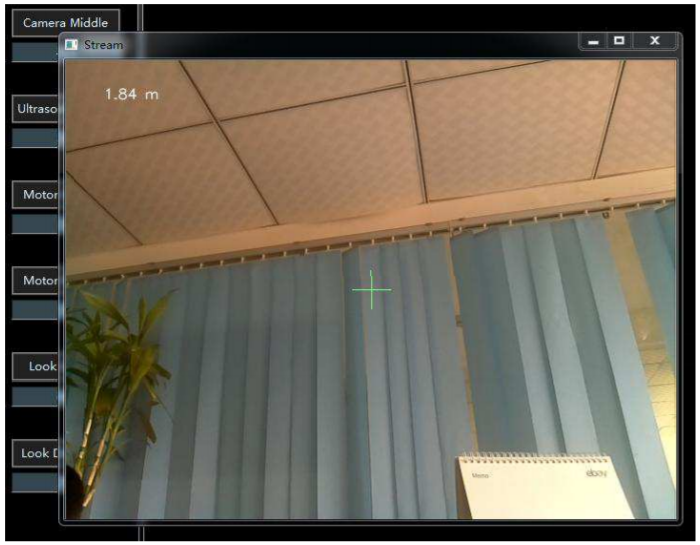
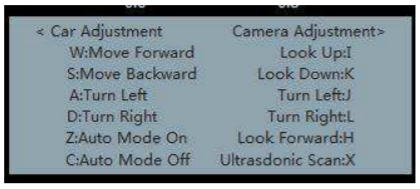
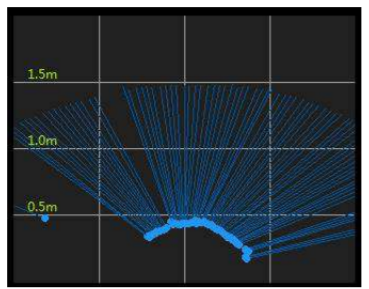
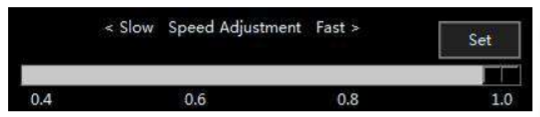
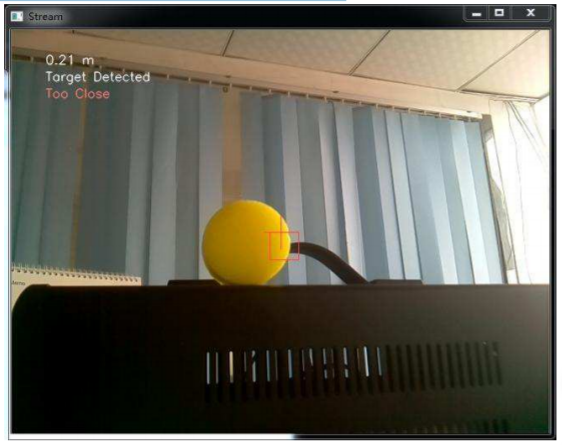
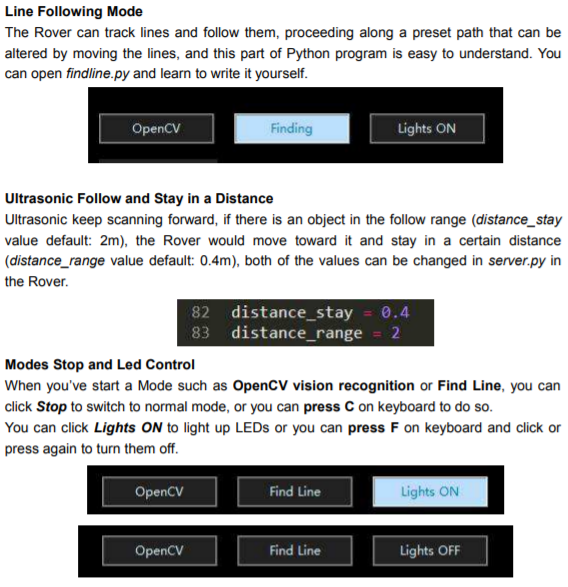
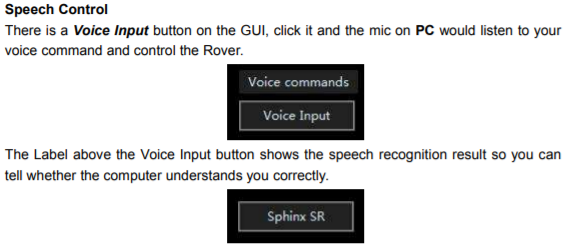
There are many instances in the manual where the description of a software or install command is disorganized and located far away from the actual number, bullet, or paragraph where the text *should* be located. The text in the Software part of the manual is clumped together and has no concept of using line spacing or white space to make text more readable. We found ourselves frequently losing our position in the manual, frantically scrolling back and forth to attempt to find what we were looking for.

Although the PiCar-B comes with its own “hat” for connecting the motor and sensors to the Raspberry Pi’s GPIOs, in the future, I would like to research how to attach more or different sensors to the Pi & PiCar to alter the functionality of the robot.  
 We ran into many problems and time constraints building and programming the robot, but once assembled, the Graphical User Interface of the software included with the PiCar-B made it relatively easy to see the direct results of the code, sensors, and user commands. Our testing methods for the PiCar, once we were done assembling and downloading, consisted primarily of using the manual’s GUI pictures to determine what the client.py program *should* be displaying versus what it is *actually* displaying. In general, the pictures, specifically in the Assembly and Testing/Running sections of the manual, were *much* better at communicating the author’s intent and instructions than whatever Google-Translated English text the authors inexplicably got paid to write.

Another problem we ran into was the PiCar turning left but not right. We could see each of the Keyboard Controls/Shortcuts worked triggering in the GUI, but when we pressed Right, the PiCar’s wheels would not turn past 90 degrees (straight ahead of the robot). We determined that the front wheels did not have the correct range of motion because the servo somehow got incorrectly calibrated. It took an incredible amount of time, effort, and questioning the existence of God to be able to disassemble the robot, recalibrate the servos, and finally put it all back together.

In addition, the manual itself is written for users with a Windows computer, which made it extremely difficult to install the necessary software on our Macbook. What seemed at first like easy-to-follow directions became nightmarishly difficult on the Mac!

**Running & Testing the PiCar-B**

* *“Switch on the car.*
* *After a while, if the LEDs turn red, it means the car’s server is connected to a Wi-Fi waiting   
   for the PC client to join.*
* *If there is no Wi-Fi for the car to connect with, the LEDs turn* ***blue****, it means the car has set  
   up an AP-Hotspot, you can use your PC to search it, the RPi Car’s AP-Hotspot’s SSID  
   name is* ***AdeeptCar*** *and password is* ***12345678*** *.*
* *Double click to run the file* ***client****.py in the folder* /client [on your PC]
* *“For initial running, you need to enter the IP address of the Raspberry Pi car IP Address,   
   then click Connect, and the program will connect to the Raspberry Pi.*
* *After connection, the program will save the IP address. For the next use, if the IP address of  
   the Raspberry Pi has not changed, you may press Enter directly next time to connect.*
* *After the connection is made successfully, the Video window shows up.”* ( Manual, pg 71) **
* *Now you may control the car by the keyboard based on the instructions on the GUI.*[note: “Ultrasdonic Scan”]  
  
* *“NOTE: If the motor isn’t working at the right direction, you should unplug the motors from A+ A- and plug it into B+ B-*.” (PiCar-B manual, pg 71-72)
* *“Press X to start the PiCar’s ultrasonic scanning. During this process, the car is unable to execute other actions. After the scanning is done, the results will be shown on the GUI as shown below:* [sic]*”*
* *“You may select the range of scanning in the upper scale and set the speed of car running in the bottom one. After all changes, you need to click* ***Set*** *to send the new data to the car.”*  
  
* Tested **OpenCV Vision Recognition**  
    *“By default, the Rover finds the biggest yellow object in its view and follows it with its LEDs turned green. When it gets close enough, it would stop and the LEDs turn blue, and if it gets too close to the yellow object, it would go back with the LEDs turned red; when the rover couldn’t find a yellow object, the LEDs turn yellow.”*
* Tested the **Line Following Module**, **Ultrasonic Follow/Stay at a Distance**,   
  **Stopping Modes**, and **Controlling** the side & bottom **LEDs**
* **Speech Recognition** & **Control**  
  
* **Safe Shutdown**  
  

**4. Use Case 1 – How the device can be used as a novel IoT device?**  
 Because the sensors, actuators, and software on the PiCar-B are so diverse, there are many potential use cases with some or all of the PiCar’s components. The technologies in this device are currently being used in systems to create things such as:

* Self-Driving, Automated Cars, which implement sensors like the Ultrasonic Ranging Module to maintain a fixed distance from the cars in front of or around the user’s car;
* Bluetooth Handsfree Speech controls for phones, cars, and other everyday systems; and
* Traffic Cameras, which use Pixel- and Object-Recognition Software to issue tickets when traffic laws are broken, or to report traffic slowdown/accident data to news stations and GPS services like Google Maps or Waze.  
    
   As these technologies become faster and cheaper to produce, they will only become more widespread within the world’s markets, with more and more real-life applications or use cases for them being created every day!

**5. Use Case 2 – A second use of the device that is different than use case #1**

Another idea I had is to use the Line-Tracking software, in addition to a GPS system, to automate Street-Sweeping Trucks and reduce the human labor cities require to keep their roads clean!   
 These trucks are typically driven (by a human worker) along both sides or gutters of a road in order to pick up leaves, dirt, refuse, and other debris. I think the line tracking software we implemented in our PiCar should easily be able to follow along a city’s curbsides, and could efficiently use a GPS to navigate through the city algorithmically by making the fewest possible number of turns. This seems like it could be a clever way to save gas, energy, human labor, and money if implemented correctly, however its feasibility would have to be adapted for different cities with dissimilar layouts, varying types of curbs/roads, et cetera.

**6. Use Case 3 – A third, unique use of the device**

A third, and decidedly less serious, use case for a remotely-controlled robot like the PiCar is to minimize human injuries and casualties in war and combat, especially by having world leaders resolve their disputes through robot battles! (<http://www.robotbattles.com/>)

As hysterical and hilarious as this idea is at first, I believe that battling robots really would serve as a better conflict-resolution strategy than large-scale wars fought by flesh-and-blood humans that have families, friends, and entire lives of their own. The PiCar-B obviously comes with no weapons of its own, (besides the frustratingly difficult manual hehe) but the systems for driving the PiCar (by keyboard, GUI, or voice), tracking objects in the PiCar’s field of view, and communicating the state of the PiCar with LEDs could all easily be reused for this context.

A brutal truth is that a remotely-controlled weapon like this could cause great harm if equipped with lethal forces to track and kill humans, but if human-waged wars continue to occur, these same bots could be transformed into a soldier’s best friend: bots that carry medical supplies and extract wounded soldiers or civilians from hostile zones that are unreachable by or are unsafe for traditional human combat medics.

**7. Conclusions about how this device can contribute to the future of the IoT**

Because we decided to follow a pre-existing kit with a pre-existing set of instructions, the current, base model of the PiCar cannot contribute anything particularly valuable to the future of the Web of Things.   
 However; with a few more modifications, sensors, and software packages, such as the algorithm to sweep all of a city’s streets in the fewest number of turns and shortest amount of time, the PiCar could become something completely novel and unique, as well as something that creates value for society!  
 For the future, I am most excited to see self-driving street-sweepers and cars (or at least some other, safer, form of transportation), but I am definitely most scared of seeing new automated weapons and tracking systems come to light.