Let's talk about Duckietown

In April 2024 we received a batch of 12 Duckiebots with NVIDIA Jetson Nano 4GB (variant DB21J) as part of the Classroom Kit (12). The main motivation behind choosing duckietown as our educational robot was mainly based on two aspects: 1. Open Hardware and documentation: The fact that it is Open Hardware gives us the flexibility (in the future) to extend the capabilities of the robot with different sensors or actuators. 2. Educational content: Duckietown offers a MOOC on autonomous driving, plus a series of exercises and notebooks called duckietown-lx as practical implementations of some concepts, this was the selling point to choose duckietown as it offloaded the amount of work required to prepare the material and exercises of a course from scratch, so it left us more freedom to focus on supporting students in their learning process instead of worrying about the material.

About the Hardware

The materials: One thing you need to know is that duckiebot is a low cost robot (compared to other educational robots), however the low cost comes at the cost of the materials the robot is made of. At first impression you receive a box containing all the materials and even tools needed for assembly, this is much appreciated. It was no surprise that the main structural material of the robot is acrylic, which despite being a bit fragile once assembled correctly feels like a solid construction (remember it is the variant DB21J). Of course it is not perfect and presents some medium and long term limitations in the life of the robot, many of those cons are listed in the paper Duckietown Project Pros and Cons, just consider that the paper review the varian DB21M, but some problems still remain (more details on this later in this post).

The assembly instructions: As mentioned above, we have the DB21J variant, so the assembly instructions are as follows Assembly - Duckiebot DB21J. It should be noted that the instructions are relatively easy to follow, but in some cases the images are not explicit enough or there is no explanatory text and it assumes that the student knows what to do with those parts, here are a series of problems that our students had during the assembly process:

- Camera Assembly: During the process of assembling the camera there is no mention of the importance of the flex orientation, it must be taken into account that many students are not familiar with this type of connector so an explicit mention or an illustration about the correct orientation of the flex would be very useful, also small warnings could be included about how delicate the pin lock of this type of CSI connector is, since in other lessons students due to lack of familiarity are often very abrupt resulting in some broken connectors.
- Computation Unit: This section starts with step number 19 indicating the placement of two nylon bolds, and step 20 jumps directly to the fan placement, omitting the mounting of the Jetson Nano on the base plate, such mounting is not arbitrary since the Jetson Nano must be oriented correctly with respect to the base plate and the USB ports must be on a specific side with respect to the plate, those details make the assembly of the Jetson a bit of a puzzle where you can assume that it is working and then realize that something doesn't fit because that step was not done correctly. Also for some reason step 20 and step 21 are the same, as are 21 and 23. Step 25 and 26 may seem very trivial, but because the manual does not mention placing the side nuts on the plate before mounting the Jetson and only mentions the front and rear nuts in step 24, but many of our students found that they could not perform 25 and 26 because they had not placed the side nuts.
- Front Assembly: This session starts well until you get to step 45 (INTERNAL PANIC), as we said

before not all students are familiar with the CSI connector of the cameras and the two red arrows in the manual without any more explicit instructions in text/audio/video format for the students may indicate to remove the plastic locker!!!! NOOO!!!! some people tried to remove it upwards “as the manual says” even thinking it was stuck they applied a little force, and even though in the Troubleshooting section it mentions about bending the flex to make contact, you can't wait until you get there to correct the error, so duckietown team if you are reading this please add notes or warnings on steps like these. Additionally the manual in step 45 does not mention anything about the orientation of the flex when connecting it to the Jetson. Another tip is to add the “Troubleshooting” blocks right in the section or step they refer to as it is easier to know that something may go wrong and having how to correct it right there is more convenient than having to go to the bottom of the page to find it.
- MicroSD cards: of the 12 robots we received 2 of them came with microSD cards that claimed to be 64GB in the marking when in fact they were only 4GB in capacity.

Interestingly the assembly instructions of the legacy version DB21M is better documented, has better explained illustrations accompanied by text, warnings, notes and covers many of the points mentioned above, an example is step 13.

About Software

Every robot requires software and mastering software is difficult. This creates a huge barrier to entry for students interested in the field of robotics, as they are faced with the need to master these technologies in order to develop software:

1. **GNU/Linux knowledge, filesystem, peripherals management, networking**. The vast majority of robots out there use linux as the main operating system, therefore every student interested in robotics should master the basic concepts of GNU/Linux.
2. **Bash, terminal tools and bash scripting**. The most common way to interact with a robot as a software developer is through a terminal, so bash skills and terminal usage is a very useful skill.
3. **Programming skills, e.g. Python or C++**. Programming skills are crucial to be able to translate ideas into an executable set of instructions for the robot.
4. **Robot Operating System (ROS)**. The de facto standard middleware; it provides a set of tools and packages that allow the reuse of components in robotics.
5. **(extra) Docker**: Runtime to run containerized software.

Requiring students to learn that stack of software tools in one semester in a autonomous robotics course only causes them to lose focus, because instead of concentrating on concepts, implementation and algorithms for autonomous systems students have to invest time in acquiring the necessary knowledge (if they don't have it) in order to gain the skills that will then allow them to perform the implementations and assignments, for example the student may have never used GNU/Linux and now they have to become familiar with it.

The problem of this monumental barrier to entry, is the one that many companies with an educational focus are trying to address, among them of course, duckietown (more about this later); where they provide a “ready-to-use” environment so that the learner can concentrate on the concepts/algorithms and not on the whole stack of software and setups that make it possible to run. As an example (I have to admit that the example is a bit of an exaggeration, but it's just to make the point): No “conventional” computer user studies the Linux Kernel to be able to use the operating system,

because the right level of abstraction is also highly appreciated, and sometimes we just need things to “just work”.

With that preamble the following is a set of good things, and some things we consider not so good about the Duckietown software stack.

Software on students' laptops: TBA

Software on the robot: TBA