ENGR101 T1 2024

Engineering Technology

Arthur Roberts

School of Engineering and Computer Science
Victoria University of Wellington

3rd project. What we are at?

- 1st project - simply follow the instructions
- 2nd - open ended. You have to think, for Challenge anyway.
- 3rd - open ended as well and on top of it we try to make it as close to real life project as we can :
  - You work in team not of your choice
  - Deadline is not negotiable
  - Get organized!
  - Write clean code

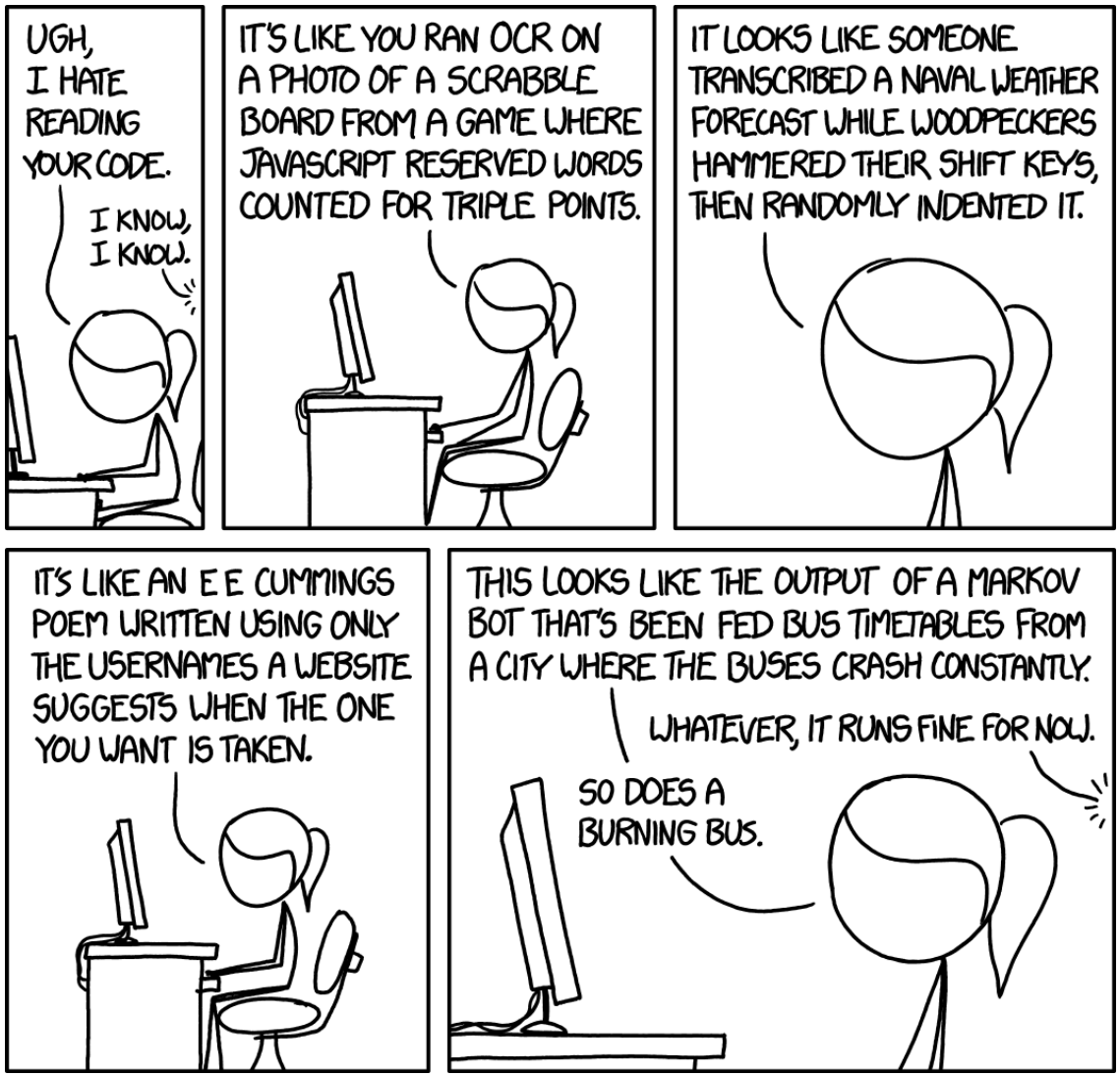
What is the drive for this project?

Your code becomes big. On average AVC project is several hundred lines of code. Writing it as one big **main()** will take long time. Other members of your will not be able to help you much. Code will be modified many times.
Motivation for this project is usual one:

- Me: Write cleaner/simpler code
- You: Why? Who cares as long as it works?
- Me: Project3 - welcome to real world. Engineers do not work alone any longer...

Compiler does not care how clean/easy to follow your code is. You just spend more time on it than necessary.
Other team members, on another hand, can get upset...

Compulsory XKCD



https://xkcd.com/1695/

Marking

AVC Project 30% of course marks 50 hrs total
Deadlines:

- AVC Project plan 2.5% same mark for whole team
- Progress Report 2.5% team mark
- Robot testing 10% team mark
- Project Report 15% individual

Marks are favoring your ability to deliver the product, organize team work and present the results your project. Robot performance is only 10%. But it is hard to write good report about non-performing robot. Report should be as technical and professional as you can make it.

AVC Project plan

We will use **GitLab** for hosting your team project.
Git in its simplest form was online repository of program files which can be shared by team members. It became much more than that.
`https://gitlab.ecs.vuw.ac.nz/` We are using ECS run GitLab server, not GitHub (privacy concerns).
First thing due is Project Plan. Make it **read.me** file in your team GitLab repository. Do not submit anything, we will pull it from you team repository.

Project plan

Roadmap

Date	Objectives	Due date	Item due	Conflicts	Tasks
...	Start project	4 June	Testing code	102 Test	**All** Complete AVC plan **SJ** Tests all installations. Write test cases for the team **BG** Ensures Plan is done. Help to install SFML on all team computers **EM** Ensure Robot is built **JB** Get/save pictures from the robot
...	Core and Completion	11 June	Core code	171 test	**All** Write code so robot moves through Q1&Q2, Intro & Background for report progress report started **SJ** In charge of Q1 code **EM** In charge of Q2 code, Checks code for Q1&2 enables robot to completes circuit, debug/fix if necessary

Project Plan should contain:

- List of team members and their contact info.
- Roles assigned to each member of the team.
- Roadmap - list of tasks with time frames, team member responsible.

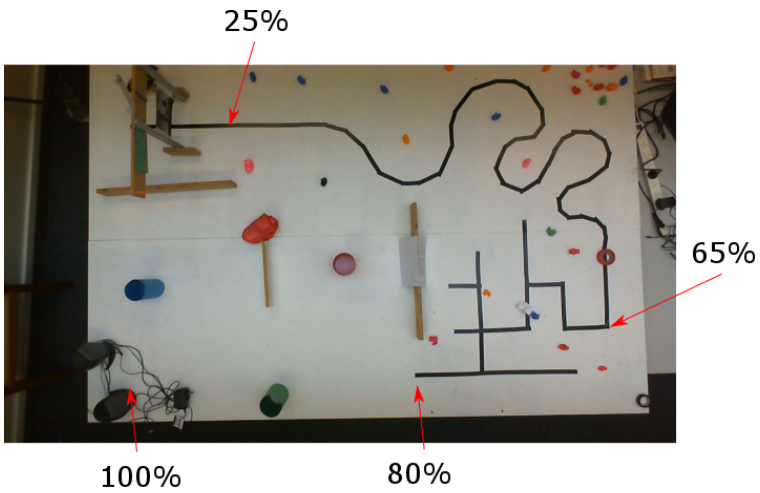
Marks based on how detailed plan is.

Progress Report

- Main purpose of the Progress Report for you to compare your team progress with the plan - wake up call.
- Roadmap is used as a base
- More on it later

Marks are based on how closely plan was followed.

Robot testing



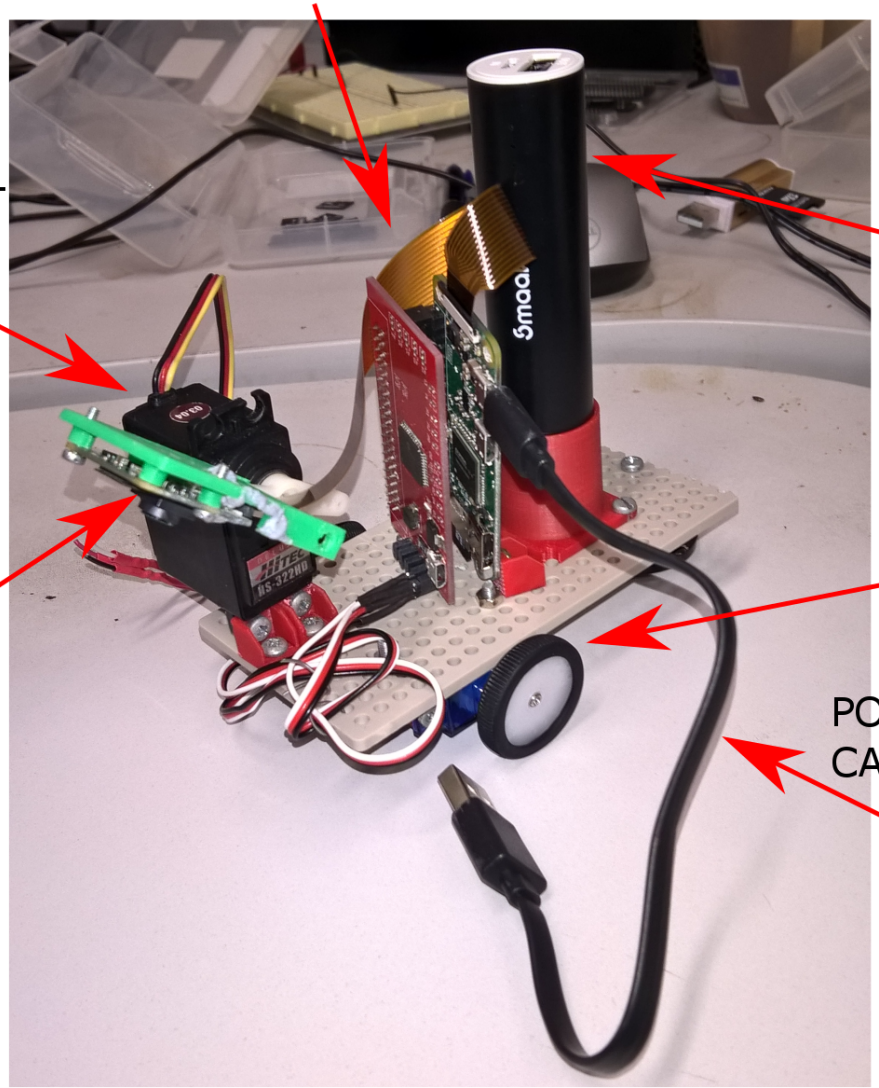
- Marks for the robot are based on how far it went on testing ground
- Quadrant 1: Open the gate by exchange with the server over WiFi
- Quadrant 2: Follow wiggly line
- Quadrant 3: Make sharp turns. Logic of direction selection - tricky.
- Quadrant 4: Came close the coloured cylinders (not hitting and toppling them) and drive to the finish line. Push ball off the table and do not fall down.

AVC report

- Industry-standard report on completion of the Project.
- There is standard structure for such reports (Abstract, Method, Conclusion, list of references) which readers are expecting.
- Marks are based on structure, clarity, presentation, technical details.
- No matter what your responsibilities as a member of the team were - describe whole Project.
- We will provide examples of good (and bad) reports.

Design 1: Really bad example

Lets start on technical stuff.. Team will have to build the robot.



Put together in 15 minutes.
What is wrong with it?
Almost everything (wheels are almost OK, though).

Design 2: Really bad example. Why is it bad?

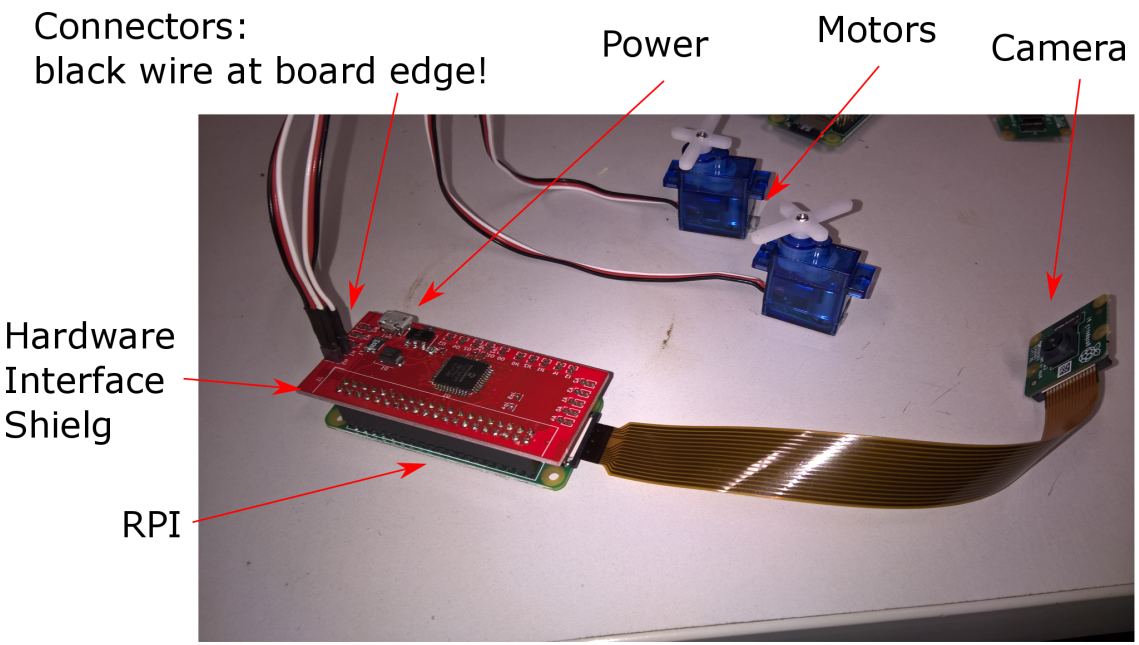
- Weight distribution. Battery is on the back - wheels are not pushed down and will be slipping
- Cables can get into the wheels and testing course obstacles
- SD card (which can break if power is switched off during writing to it) is impossible to change without taking robot apart.
- Keyboard and mouse can be connected easily - good part.
- It looks ugly

Design tools

- Meccano-type plastic pieces
- Wheels of different diameter
- 3D printers. You design and manufacture parts you need: brackets, holders, etc.
- Computer-Aided Desing software. You can use any you want. 3D printers take stl format files We provide support for FreeCAD - installed on ECS computers.
- FreeCAD demo - use "Part" workbench. Basic shapes, parameters, boolean operations

Hardware: RPI Zero W with custom electronics

Same as Project 2 except that motors are added.



- Take image from the camera
- Decide what to do
- Control motors
- Power: from PC USB or power bank(provided)

Figure:

Motors should be connected so that white wire comes on the top.

How to program moving robot without keyboard, mouse and display (headless)?

We connect to it over WiFi network. How do we know which computer connected to network is our Raspberry?

Just as all program variables are assigned a memory address, so all computers connected to the network are assigned addresses. Internet Protocol (IP) address.

RPI: type in **ifconfig**. Result is something like: 192.168.1.72, as an example. Each field is char (0-255). So, altogether we can have 255 4 = 4, 228, 250, 625 computers connected, which is not enough.
IP of 192.168.1.72 is IPv4 address. Move to IPv6 is underway. IPv6 is fe80::9f03:802e:3e1a:aa50 (notice hexadecimal notation) and can address 3.4x10 2 8 computers.

How to program moving robot without keyboard, mouse and display (headless)?

Now that we know unique IP address of the RPI we can connect to it.

- On lab computer type in **pi@192.168.1.72** (replace the number with the IP address o)f RPI. **pi** is default username.
- Reply **yes** in dialog box
- Enter defaul password **raspberry**
- You are connected. Terminal(CLI) window is running RPI, not lab computer
- To disconnect press **Ctrl-D**

Terminal common commands

- **sudo** - get administrator permissions. You need that to run your program.
- **ls** - list all files and directories in current directory
- **cd name** - move (change) to directory **name**
- **cd..** - move one directory up
- **mkdir/rmdir name** - create/remove directory **name**
- **rm name** - remove file **name**
- **./name** - run program **name**. For this project it should be **sudo ./name**.
- **nano** - starts command line text editor

Compiling C++ file without Geany

When connected to RPI over ssh - only terminal is available.

- To compile file **f1.cpp**: enter **g++ -Wall -c f1.cpp**
- **-Wall** enables all warning messages
- To build executable **f1** from object file **f1.o**: enter **g++ -Wall -o f1 f1.o**
- If code needs library: **g++ -Wall -o f1 f1.o -le101**
- Shortcut - to compile and build in one command: **g++ -Wall -o f1 f1.cpp**
- To run the executable: enter **./f1**.

Make files

- Typing `g++ -o example example.c` all the time wears the keyboard and your fingertips
- There is a shortcut - makefile
- Format of the **makefile**:

Listing 1: makefile

```
m3: m3.o
        c++ -o m3 m3.o
m3.o: m3.cpp
        c++ -c m3.cpp
clean:
        rm *.o
```

Listing 2: makefile structure

```
target: prerequisites
        TAB command
```

- To execute makefile type command **make**.

Program structure

- Once, at the start of the program: call init(0) - it initializes the hardware
- Run forever loop: while(1) Inside the loop:
  - Take the picture (take_picture()) You don't need to display the picture - makes RPI to go faster.
  - Do video processing. It is hard/impossible to use same algorithm for different quadrants.
  - Adjust motor speed based on results of video processing
- Stop the robot and terminate while loop after pushing ball off the table.