RD - MTRN4110 – Project 1 / S1.2017

PREPARING COMPONENTS AND DEVELOPING NECESSARY MODULES

This project involves developing a number of modules and components, which will be used in subsequent projects. The performance of these modules will be relevant for those subsequent projects. This fact means that you should invest proper effort, not just for successfully passing this project, but for producing good solutions, because you will use them frequently.

This project is divided in a number of parts, for preparing the on-board computer (OBC), for interfacing with sensors and for performing certain basic processing tasks.

We assume that the students (who should be, at least, in year 4th, Engineering) do have sufficient knowledge, acquired in previous courses, for solving the programming components, and also have adequate general knowledge and common sense, for solving the diverse issues involved in this project (connecting computers, using power supplies, configuring networks, installing/configuring 3rd party software and drivers, using C/C++ compilers, using Matlab, proposing validations procedures, reading manuals, applying Mathematics and Control concepts, etc.)

All the parts, which are required in this project, are assumed to be team work, except the ones that are declared, explicitly, to be INDIVIDUAL assignments.

Part 0: Prepare your On-Board Computer. (point marks 0/12)

Each team will be assigned a small computer, which will be used in your projects, for implementing the on-board processing and intelligence of your robot. We refer to it as the "On-Board Computer" (OBC).

You will need to perform some settings and tests to make the OBC fully operational for subsequent applications.

You are required to configure the OBC for being able to be accessed remotely, via remote desktop software.

The provided OBC already has certain predefined settings and installed software modules:

1) VNC server, already installed and configured (listening at port=5900, password=mtrn4110)(*).
2) Unique IP associated to its Ethernet adapter. This allows your OBC to operate in the Lab's LAN, via Ethernet, not having conflicts with the rest of computers (Lab's desktop PCs, other teams' OBCs, dynamic range of IPs for students' laptops). The IP assigned to your OBC is unique in that LAN.
3) WiFi, to connect to your home Wifi or Laptop.

The OBC can be connected to via a WiFi connection. A recommended approach is to use a freely available Virtual Router program that can be installed on your own laptop for sharing the current WiFi network you are on.

https://virtualrouter.codeplex.com/

It is recommended that a fixed IP is selected for use with the Virtual Router program, but it is not necessary. The virtual router allows your laptop to create a new network offering a separate SSID (and associated password); consequently, you can configure your OBC to automatically connect to it, so that you may use it without needing to connect the Ethernet cable.

You may change those settings for working in other places (e.g. using your Ethernet switch or WiFi at home), however, when you use it in the Lab, you must use the predefined settings, to avoid conflicts.

Alternative ways of connecting with your OBC:

1) Point to Point:

You may connect to the OBC, just using an Ethernet cable, for a "point to point" connection to another computer (e.g. your laptop). In that case, you will need to set a fix IP number, for the Ethernet adapter in your laptop, having the same subnet (192.168.1) but a different number.

Subnet mask: 255.255.255.0

IP: 192.168.1.XXX where XXX should be different to the one of your OBC.

(note: Do not use static IP numbers in the Lab LAN, if you connect your laptop. Change its Ethernet adapter's settings to have an IP dynamically assigned.)

2) Use the LAN, in the Lab. Your OBC can be connected to any of the available Ethernet cables. Your OBC has a unique IP (we assigned it); consequently, no IP conflicts will occur (in that LAN). You can use any of the PC's in the Lab for communicating with your OBC. If you use your laptop, its Ethernet adapter must be configured to dynamically obtain IP number (as mentioned in (2)). If you use a desktop PC, you do not need to modify any setting of that PC.

3) Using other networks, e.g. WiFi at home.

You will need to set the OBC's WiFi for automatically connecting to certain preferred WiFi networks; so each time you turn it (your OBC) on, it will connect to that preferred WiFi, if that one is present/detected. You need to set the WiFi settings, in your OBC, to assign IP according to your router settings (usually dynamically assigned). However, you will need to infer it, when using a client remote desktop. In addition, connecting to a public network may imply some security issues, so you may need to set the OBC's firewall properly, in that case.

The OBC 's operating system (OS) is Windows 10; you can (and should) modify certain settings to make it secure and adequate for your applications. It is assumed you have experience in managing basic matters in such type of OS. If not, you should learn them, e.g. from the Web.

4) Discuss with the Lecturer, or with the leader tutor, for other alternatives you may prefer (e.g. changing OS, etc.).

Note: If for some reason you "destroy/corrupt" the OS image, in a way that the OBC does not start anymore, we can reinstall the OS image again, for a limited number of times.

All the steps, included in this part of the project, are typical matters which a person, having certain medium technical knowledge for dealing with computers, should have. If you do not know how to do certain parts, you will investigate (e.g. from Web) and learn yourselves. We do not teach these matters in MTRN4110. We recommend or request them to be used.

We may help you, for solving unresolved issues; after you tried to solve them. You are a team of engineers; consequently, we expect you to work as a team of engineers.

(*) You may change the password (recommended)

Part 1: (Points 2/12)

Implement a small client-server system, for communicating remote applications (e.g. from your laptop) with the OBC, via TCP/IP.
You choose the TCP protocol you prefer (e.g. TCP or UDP). You may reuse/modify source code you have implemented in previous subjects (e.g. MTRN3500), provided that it is yours, that you understand it, and that you are able to explain it in your demonstration; in case you are asked about it.

The programs (client and server) should offer the following basic capabilities.
1) From the client side a simple program (e.g. in console mode) will allow users to specify some basic commands, to be sent to the server. These commands will be the following ones:

a) Turn ON/OFF server's service.
b) Set "frequency" of messages/samples.

The server, which will usually run in the OBC (or any other computer, for testing purposes), will send, periodically, certain virtual (synthetic) measurements. Each measurement will be composed by an array of 10 numbers of type *unsigned short int* (i.e. 16 bits), which will be increased (by the server) each time they are sent (note: for software validation purposes, they should be time varying).
The rate of "sampling" / transmission, will be defined by command (b), and the activity by command (a) or by inferring it from the specified frequency; e.g. if F=0 HZ, the command can be interpreted to be "STOP transmitting". For (b), the maximum allowed frequency will be 50HZ.

Note 1/assumption: you can assume that there will be ONLY ONE client. There is no need to support multiple simultaneous clients. This assumption will be valid for this project and the rest of the projects, if properly implemented.

Note 2: you can start to work on this part, even before you receive the components (OBC, 3D camera).

Demonstration: your client program will show the activity (displaying/printing a counter for indicating the current number of received data messages, from the server). The evaluator will operate your client program, to verify its operation. No further evidence will be required for this part, during the demonstration. However, the evaluator may ask you to show details of your implementation, if he/she considers it necessary.

Part 2: Using the 3D camera (point marks= 3/12)

2.1) Implement a small program, for acquiring depth images (from the RGB-D camera), locally. For "locally" we mean that the program runs in the same computer were the camera is connected (via USB). There is an API (i.e. a library) for reading images from the camera (the API is available on-line, from the maker's web site; it will be also offered via Moodle). Use the camera in a low resolution mode (and subsample the raw images, if necessary). Resulting depth images should be 120 (vertical) by 160 (horizontal) pixels.

Demonstration: you do not need to demonstrate this part, if the subsequent part (2.2), is demonstrated to work.

If that is not the case, you will be required to show the code (of this part) to the evaluator.

2.2) Adapt (2.1) for allowing remote sensing; in this case the program will run in the OBC (the camera will be connected to it), and it will also operate as a sensor server, providing periodic measurements to the remote client. This program is a combination of the capabilities of items (1) and (2.1).
The transmitted frame rate (which is defined by the client) will be limited to Fmax = 2 HZ (consider that fractional frequencies, e.g. F=1.5HZ, are also allowed.). The user specified sampling frequency is approximate; it is not required to be exact.
To show that the program is working, you may show images in a window, or save one of them in a binary file when requested by the user. This way of showing that the program is working is superseded by part (2.3); if that part is working, it will be considered that your part (2.2) is operational.

2.3) Show the sampled images, in an ON-LINE fashion, in the remote PC, using Matlab or other programing context. A Matlab's client is recommended because it allows, easily, implementing TCP communication; and also offers rich resources for processing and showing images.
However, you may implement your client application in other programming languages, such as in C++ (as you did in MTRN3500) or in Java, Python or others; provided that your program offers the required capabilities.

The relevance of each component of this problem is one (1) point each, totalizing 3 points of the 12 of the full project..

Part 3: Processing 3D data (point marks= 2/12) (this is an individual assignment)

Implement a program for processing 3D images, for inferring the presence of OOI's (Objects of Interest), nearby the sensor. Focus your processing, on pixels located "over the horizontal", to avoid false alarms due to detection of the floor. This assumption will be valid when the camera is located close to the floor (e.g. on a hexapod) and facing ahead, horizontally or having a small inclination in pitch (which will be known by you, because you installed the camera).

You are required to propose and implement your approach. The program is not specifically required to work locally or remotely (you choose), but it needs to run in an ON-LINE fashion, at a processing rate of at least 1HZ. This program can be based on Part 2 (e.g. 2.1), adding the processing part (which is required in this part); alternatively, you can use your private version of the image acquisition module, in place of the one developed in (2) by your team.
An OOI is defined to be any vertical (or almost vertical) object whose apparent size (e.g. diameter) is at least 3cm, and smaller than 50 centimeters, and whose approximated distance to the camera is lower than 1m.
You may perform this inference process by processing *certain* horizontal line of the image, and by assuming that all the objects are prismatic (in the vertical dimension). More complex and powerful inference can be achieved by processing the full 3D image (or at least using more than one horizontal line); **however**, such an approach is not required in this task, although you are allowed to try your own ideas, if you want.

The program will operate in the following way.
1) It will read camera images (e.g. as in part 2.1).

2) The program will process images, at a rate of about one image every second (even if the camera produces higher rates).
3) It will show/print the result for each of the images being processed. The estimated (approximate) position of the OOIs, will be reported by simply printing the 2D positions of the detected OOIs. Alternatively, those estimated positions (i.e. points) can be shown in a figure, e.g. as usually done in Matlab (MTRN2500) or as you have done in C++, in MTRN3500. Those estimates must be consistent (i.e. each of them should be "close enough" to an apparent OOI).

Demonstration: The evaluator will use two or three small pipes (about 15cm tall, 5 cm diameter), which will be located at different positions, on the floor, appearing like poles, in front of the camera. He/She will verify that the detection and inferred positions of those OOIs are consistent.

Assumptions:
a) Your program is expected to detect a variable number of OOIs, however you can limit that number to be no more than 10 OOIs.
b) In case of partial occlusions, the partially occluded OOIs are not strictly required to be detected. Similar consideration is valid for the cases of OOIs which are located at the boundaries of the images.

Recommendation: for testing your approach, you may save some images, and try processing them in an OFF-LINE way, before trying the ON-LINE version of your program.

Part 4: Reading AIs (Points 2/12)

This program will periodically read certain Analog Inputs of the OBC's embedded Arduino. As the OBC does not have sufficient IO capabilities, we exploit its embedded co-processor (i.e. an Arduino) for those purposes.

The structure of this solution involves the following components:

a) Arduino program for reading three (3) AIs (e.g. AIs 1, 2 and 3) and for sending the data to the OBC, using the built-in serial port (which connects the OBC's main CPU and its embedded Arduino).

b) The OBC sends those measurements to your remote PC. In the remote PC: use Matlab (or a C/C++ program) for sending commands to the OBC, and for receiving command responses and data, from the OBC. Show the measured values (simply printing them or, if you want, by other way you consider appropriate, e.g. simulating an oscilloscope.)

Note 1: You can see that some parts of this program were already solved in previous parts of the project. The new part is related to using the Arduino (in a basic way, i.e. just periodically reading AIs), and sharing those readings with the OBC, which eventually transmit them to the remote client.

Note 2: certain basic sensors produce measurements which are easily read via ADC (Analog to Digital Converters, aka Analog Inputs). Some of those devices are the IR and US range sensors, which you could try in this project, but are not strictly required in it.

Note 3: For testing your program you may replace the AIs' readings by using simulated values in the Arduino program, or by using potentiometers (of **adequate values**, for avoiding improperly loading the Vcc terminals of the board) for generating variable voltages on the AIs. During the demonstration, the evaluator of your project may use his/her potentiometer for testing your program.

Note 4: you can start to work on this part, even before you receive the components (OBC, 3D camera).

Part 5: Reading and Processing IMU measurements (Points 3/12)

This program is very similar to the one in Part 4; however, in this case, an Arduino program will read IMU measurements, at a frequency ~50Hz. The reading will be shared with the OBC (similar to Part 4).
As during the first weeks of this teaching session, you will still not be able to use the real IMU (till you are allowed to perform some soldering), you will simulate the measurements of the six channels (3D accelerometers, 3D gyroscopes).
The pattern being simulated will be a trivial periodic sequence (e.g. steps), which will be specified by your session's tutor or by the lecturer, during the lecture.

Part 5.1: (0.5 points)
Simulate the periodic pattern in the embedded Arduino, transmit the values (@50HZ) to the OBC.

Part 5.2: (0.5 points)
This part implements the OFF-LINE integration of gyroscopes, for estimating attitude.

a) Mode 1: Integrate gyroscopes in 2D (estimate Yaw, assuming 2D operation)
b) Mode 2: Integrate gyroscopes in 3D (estimate Roll. Pitch and Yaw, assuming 2D operation)

You are recommended to use, for this part, Matlab.
In both cases (modes 1 and 2), plot the resulting attitude at the end of the processing.
Use the proposed periodic pattern, for a number of periods, for testing it. You can compare your results with the provided solutions (to be provided by the Lecturer, in week 2).
Demonstrations of (5.1) and (5.2) are superseded by demonstration of (5.3), if this last one is successful.

Part 5.3: (2 points)
This part implements the integration of gyroscopes, for ON-LINE attitude estimation. This program will process the data produced by the Arduino (implemented in part (5.1)). The processing will follow the approach applied in part (5.2).
Similarly to (5.2), in this part you will offer two options:
Mode 1: Integrate gyroscopes in 2D
Mode 2: Integrate gyroscopes in 3D

This program can be implemented as an on-board process (OBC) or as a remote one (e.g. in Matlab, in the remote computer).

Note 1: the mathematical approach, used for integrating the angular velocities is described in the lecture on week 2.

Note 2: About the scaling and numerical representation of the data. All the measurements (3 accelerations and 3 angular rates), will be represented by 16 bits signed integers. For the accelerometers, 1 count will represent $0.001 m/s^2$; for the gyroscopes' measurements, 1 count will represent 0.01 degree/s.

Note 3: this program may be used in subsequent projects, for processing real measurements from the IMU's gyroscopes.

Note 4: you can start to work on this part, even before you receive the components (OBC, 3D camera).

Explanation: This project will be explained, during the lecture, on week 1.

Demonstrations: This project will be demonstrated on week 4, during your lab session.

Submission of files: No report and no program submission will be required for this project.

Quiz: The Quiz will take place the same day of your demonstration (in your lab session). It will run from minute 15 till minute 30. You are expected to attend it, even if you give your demonstration in a different week or day, in case you were given permission to a late demonstration.
If you are not able to attend, contact the lecturer or apply for Special Consideration.

Questions: Via Moodle's Forum or by asking the lecturer by email (j.guivant@unsw.edu.au)
