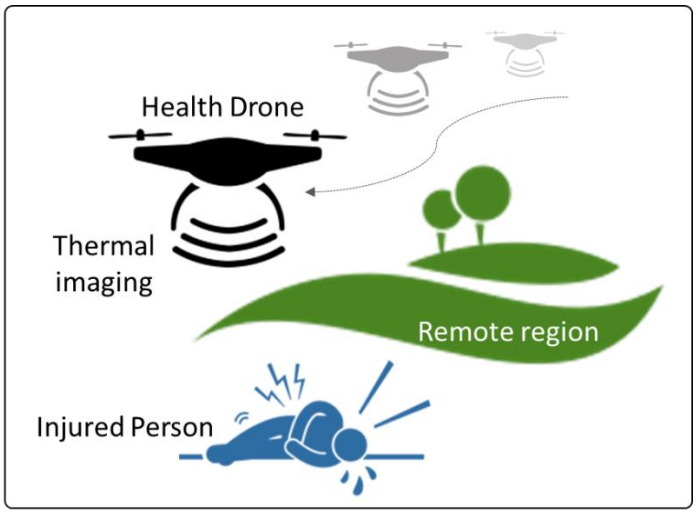
Mech1010 Coursework – MCU Systems

Global Health Drone: Autopilot Controller



INTRODUCTION

Global Health Drone is designed to travel across large landscapes in remote and resource-limited parts of the world – areas where traditional healthcare systems struggle to provide assistance, particularly during emergencies.
The drone's job is to search across large landscapes, locating potentially injured person(s) via camera systems and then use a thermal imaging system to scan and provide vital medical feedback to medics who can assess and come to help…

So your task is to program a balance control system for the drone which holds it in a stable horizontal position while the system performs a thermal scan….

You
- Lab Notes
- Practice

Simulation
- Testbed for code
- Unlimited attempts

ReLOAD
- Real Hardware
- 50 Attempts!

Please read this coursework guide carefully for full instructions, including your objectives, technical background and a mark scheme. Over and Out.

Revision 02 – 02.04.2023

AIM AND OBJECTIVES

The aim is to develop a balance control system which moves your drone to a stable horizontal position and holds it there for 5 seconds while it performs a thermal scan...

APPROACH

Drones are expensive... so you will need to develop your code carefully in two stages using:

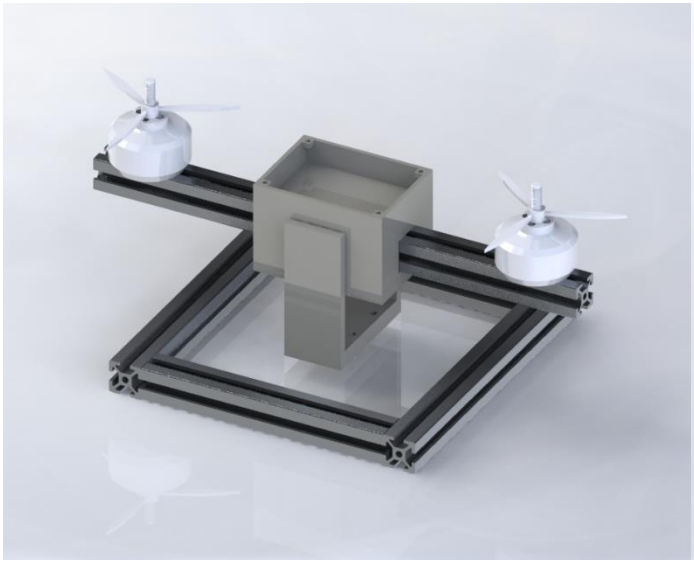
1. **A drone simulation (in WokWi):** develop your code until it is functional and error free ...
2. **The ReLOAD drone model:** now test your code on a real system and tune the controller

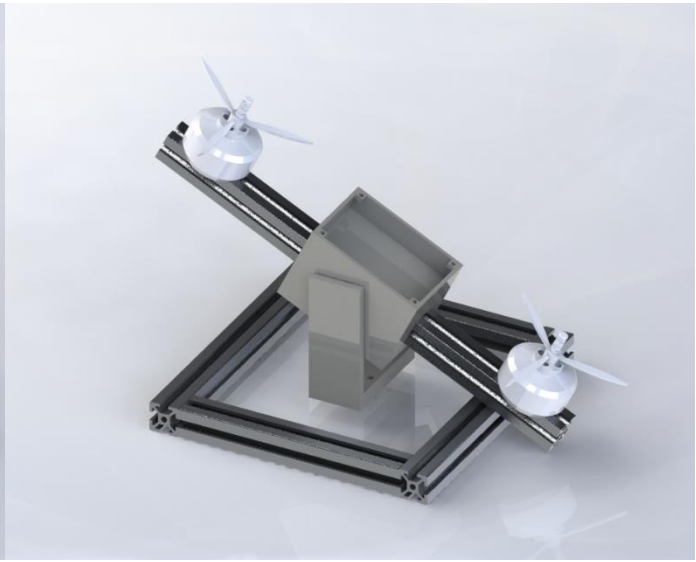
IMPORTANT: You are limited to 50 test runs on the ReLOAD system!

(This represents industry practice – testing on hardware is costly ... only attempted when you have a system you are confident will work effectively!

OBJECTIVES

1. Produce a Flow Chart of your program from the Specification (see Page 4)
2. Write an Arduino drone controller program (based on your Flow Chart) which:
    a. Initialises your system hardware
    b. Implements a closed-loop controller to reach and maintain a horizontal position
        i. Your controller should run at 25Hz
        ii. As a minimum you should implement a Proportional controller
    c. Lights LEDs to indicate (System start, Reached Target Angle (Scanning Start), Scanning complete
    d. Sends flight telemetry data via Serial – see Page 4 for the detail
    e. Use good programming practice to produce a well-structured and understandable program
    f. Use functions to implement
        i. Sensor measurement
        ii. The controller
        iii. Other functions of your choice...
3. Collect and submit evidence from ReLOAD of your system in action (see Deliverables, Page 6)
    a. Your flow-chart (1 side of A4)
    b. Your program
    c. ReLOAD outputs (.CSV and Video file)





SYSTEM DESCRIPTION

You will develop your controller using the model drone system shown in Figure 1.
The drone is represented by a beam, pivoting at the middle, with a motor-driven propeller at each end. The beam is free to rotate about the pivot, simulating the drone moving in the air. Your Arduino controller is connected to:

1) A potentiometer which can be used to measure the angle of the drone (relative to ground).
2) Two H-Bridge motor controllers, each linked to a high-power DC motor and power supply

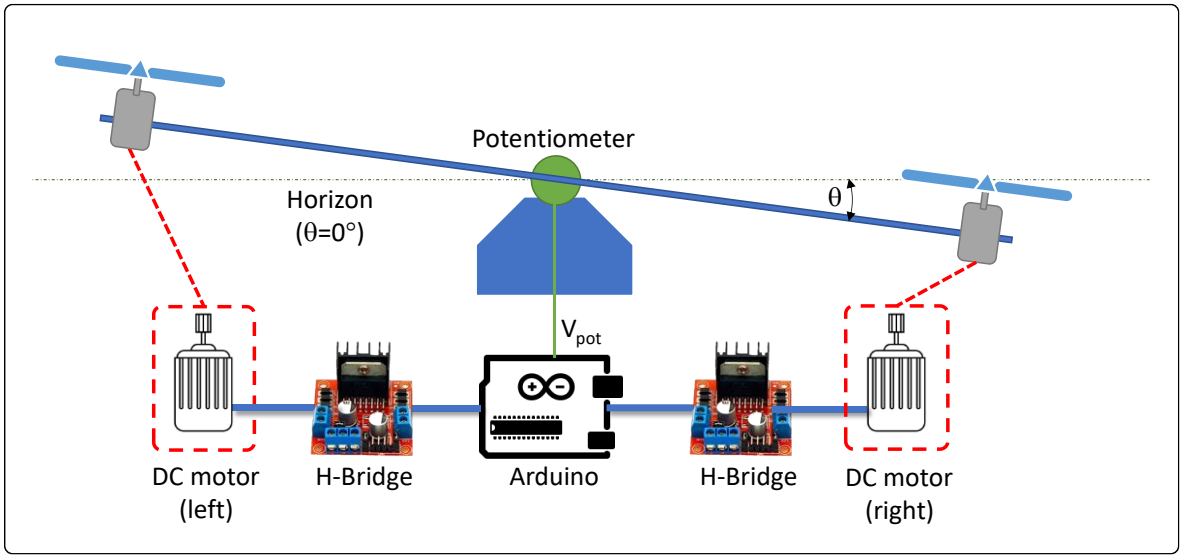


Figure 1. Configuration of the model helicopter system

HARDWARE

The hardware in the drone model is described here. You will need to use this to configure your Arduino system correctly, and to calibrate your sensor measurement.

Item	Information
H-Bridge Controllers + Motors **NOTE:** **You will control these motors using code provided in the template (See WOKWI Simulator Section) like this:** **motor_Left.write(60);** **motor_Right.write(85);** **The control signal for each motor ranges from:** **45 = Motor Off** **85 = Full Power**	A commercial H-Bridge controller. This takes a 9-24V supply **Left Motor Controller** **Control = Pin 10** **Right Motor Controller** **Control = Pin 9** **Direction** = Both Left and Right are Fixed to positive thrust
Potentiometer For calibration, we took the following measurements: V_{pot} / Angle 552 / 0 729 / -22 371 / 22	The potentiometer output voltage will change linearly as a function of the angle θ shown in Figure 1. The output V_{pot} can be measured using Arduino **Pin A0**
Output LEDS	Use these LEDs to signify the status of the control system. **Start-up LED (Green) = Pin 2** **Scanning LED (Yellow) = Pin 3** **Scan End LED (Red) = Pin 4**

Table 1: Equipment specification for the model helicopter

SPECIFICATION

Your balance controller should be written to go through the following process – please read this carefully!
You will need to convert this into a Flow Chart as part of your submission

The process is defined by a series of numbered stages. Each stage requires you to perform some tasks which are detailed.

1. As your control program starts it must initialise the system. Setup your Serial communication so you can send messages. Configure your hardware connections to the sensor and motors. Initialise the motors by sending a 0 (zero) signal to the LEFT and RIGHT motors.
2. Show that your balance controller (the programme you write) has started by lighting the Start-up LED for 1 second
3. Your controller should now start. It should run continuously at 25Hz.
    a. The controller first reads the angle sensor to determine the current angular position of the drone
    b. Now your controller should calculate a motor control signal
    c. Convert the motor control signal into control signals for the LEFT and RIGHT motors
    d. Send telemetry information using Serial communications
    e. Check if you moved within 5 degrees of the target angle (e.g. horizontal)
        i. Yes? Start your `Scan Timer` and Light the Scanning LED
        ii. Once Scanning has started – it must continue regardless of the drone angle
    f. Continue the controller until your Scan Timer reaches 5 seconds
4. Shutdown: Light the Shutdown LED for 1 second and turn-off the motors

AUTO-PILOT BLACKBOX DATAFILE

As your program runs it should generate a BlackBox datafile - formatted in the same way as the example below but with your own data.

This must be submitted as part of your coursework to demonstrate that your program runs and how well it performs!

(Note the ... just indicates lots more data!)

```
0.System Started
1.System Initiated
2.Controller Starting
Time,Angle,Error,Control Signal,Motor L, Motor R
0.00,0.3,0.15,45,55,45
0.02,0.29,0.14,41,65,50
.....
8.00,0.151,0.01,10,70,20
4. Shutdown
```

UNITS: Please ensure you report your data in the following format

Time	seconds	2 decimal places
Current Angle	degrees	2 decimal places
Error	degrees	2 decimal places
Control Signal	n/a	
Motor L	0-100 percentage activation	
Motor R	0-100 percentage activation	

Example data that should be printed to Serial as your program runs.
Note: the numeric values are only examples – yours should use the actual values you record!

THE CLOSED LOOP CONTROLLER

CONTROLLER TYPE

Your program should implement a closed-loop controller.
The basic version you should attempt is a Proportional Controller – see lecture notes/slides

For additional marks you can implement a more advanced PID controller – see links below
This must be made clear in your code comments!

CONVERTING YOUR CONTROL SIGNAL INTO MOTOR SIGNALS

Your controller will generate a control signal that must be used to drive your two drone motors and so control rotation.
Each motor can only produce positive thrust – you control the amount of thrust using a control signal to the motor (see notes)

Important: A suggested method for converting the control signal into two seperate signals to drive the motors is shown below.

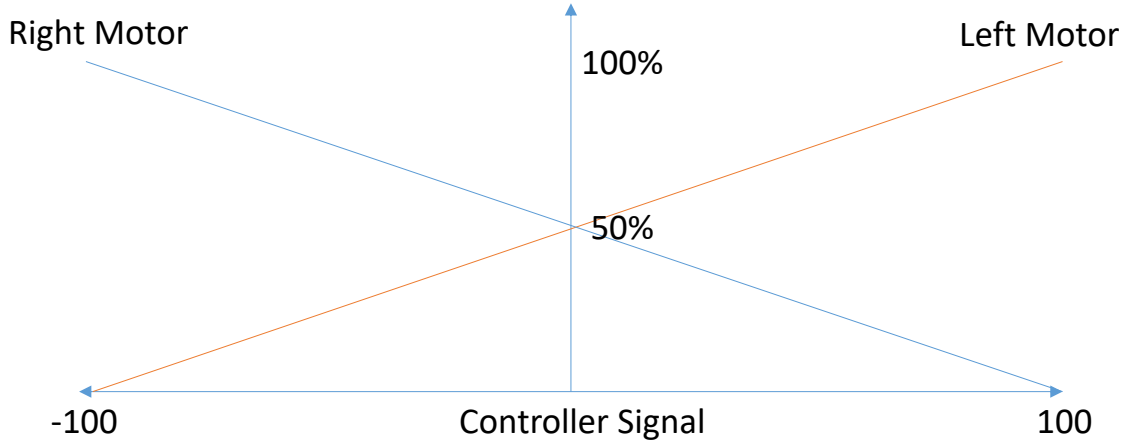
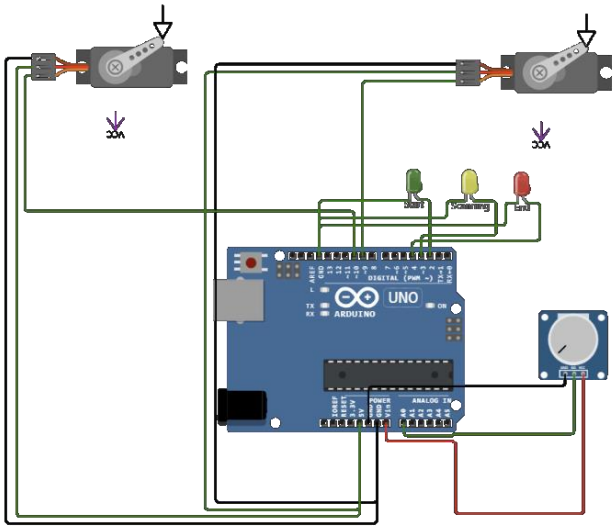


Figure 2. Method for converting the control signal into the motor command signals

SIMULATION

Please find the simulation and template code at this address:
https://wokwi.com/projects/327380710302155348

Item	Purpose
Blue LED	Left Motor
Orange LED	Right Motor
Green LED	Startup LED
Yellow LED	Scanning LED
Red LED	Shutdown LED
Potentiometer	Angle Sensor
Template Code	Your starting point!



Note:
The power output of the left and right motors is shown by:
BLACK arrow = LOW POWER
PURPLE (VCC) arrow = HIGH POWER

REFERENCE: PID CONTROLLERS

https://scholarscompass.vcu.edu/cgi/viewcontent.cgi?article=5737&context=etd
https://www.omega.co.uk/prodinfo/pid-controllers.html

COMPLETING THE COURSEWORK

HOW IS IT ASSESSED?

The coursework is worth 50% of the overall Mech1010 module mark.
It is designed to be completed individually, without any collaboration with other students.

Expertise: The programming and skills required to complete this coursework relate directly to the learning objectives of this part of the module. They have been covered in the lecture and lab activities – please refer back to these for examples.

Deadline: Please see the deadline on Minerva

Deliverables: *You must submit the following items through Minerva – correctly named –to receive a mark*

- ***Each file named in this format:*** ***username_mech1010_.XYZ***
  - *A Flowchart PDF* *username_mech1010.pdf* *(Use the provided PPTX template!!)*
  - *Arduino File:* *username_mech1010.ino* *e.g. men18xyz_mech1010.ino)*
  - *ReLOAD CSV Data* *Original name* *e.g. Experiment_1614027456688_data.csv*
  - *ReLOAD Video* *Original name* *e.g. RELOA_15_Mar_2021_14_49_5722.mp4*

CHECK: *The ReLOAD Data files are unique to your username and will be checked against the ReLOAD Server*
Please do not alter data-files or your submission will be investigated for plagiarism!

MARK ALLOCATION

The table below details how marks will be assigned for your project work. These are linked to the learning objectives that this coursework has been designed to assess. These are based on the material you have covered in the lectures, labs and self-study exercises, look back at your notes for examples and reminders!

Objective	Mark %
1. Flow chart	15
2. Angle measurement	10
3. Controller implementation	15
4. Main auto-pilot program **a. Features** **b. Logic and Process**	**25** (15) (10)
5. Formatted telemetry data	10
6. Good programming practice	15
7. Performance of the controller	10

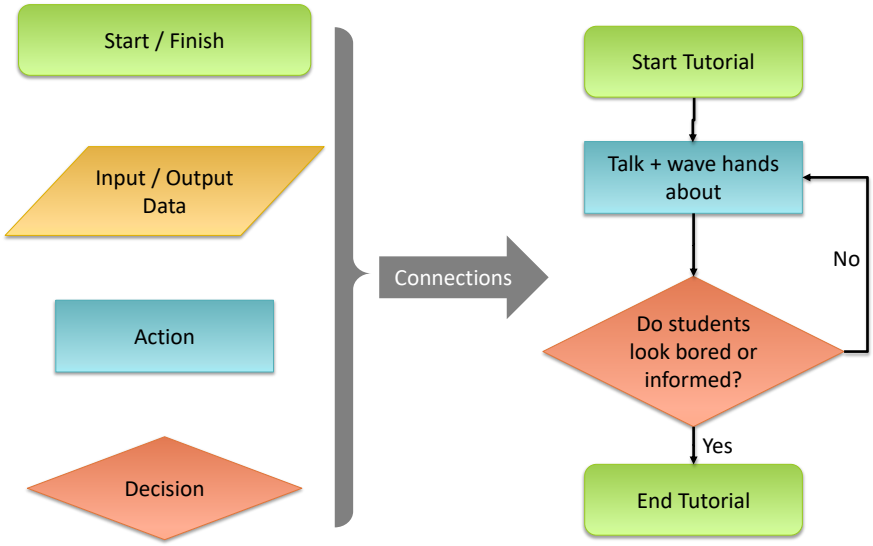
PLAGIARISM

Your code will be checked for plagiarism and malpractice – compared to other student's work.
PLEASE TAKE NOTE – COMPLETE THE COURSEWORK INDIVIDUALLY. DO NOT SHARE CODE OR DISCUSS YOUR SOLUTION!

APPENDIX: DEVELOPING FLOWCHARTS TO SOLVE ENGINEERING PROBLEMS

FLOWCHARTS OVERVIEW

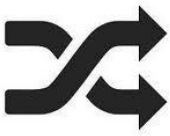
A flowchart is a graphical way to present a process or program. It describes the process from start to finish, including the key steps that need to be undertaken and any decision making and/or logic. Flowcharts are made up of a number of different shapes that represent specific actions; arrows between the shapes represent the flow of control through the process rather than the flow of data.



Creating a flowchart is a good starting point when designing any kind of program or process as it can help define the architecture of your code and pin-point issues early on in the design.

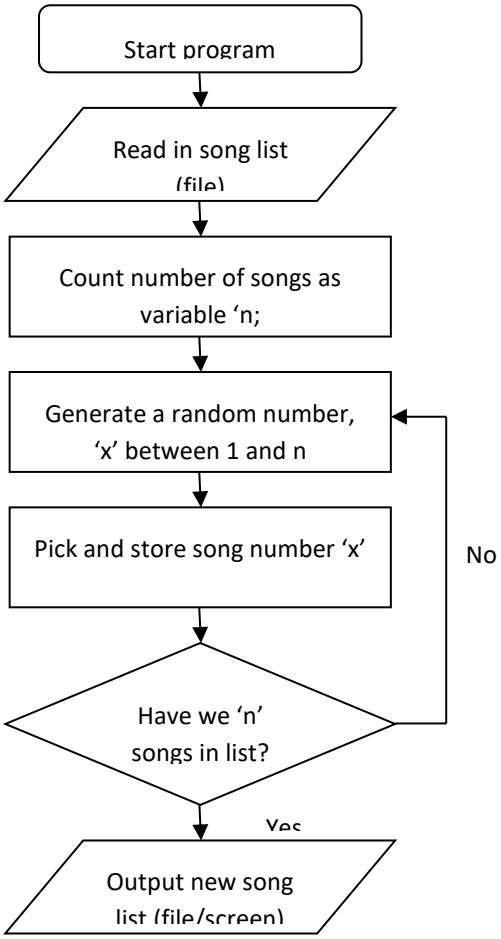
FLOW CHART – MP3 SONG SHUFFLE EXAMPLE

An example flowchart for a simple song shuffle program is shown in Figure 1. Go through it with just a few songs to see how it would function. Now consider:



- Will this program do a suitable job of shuffling the songs?
- How could it be altered to better fulfil the user's requirements?

(*you'll do this example in Matlab, Semester 2 of Mech1010*).



DESIGNING A FLOW-CHART

When producing a flowchart follow these guidelines:

- keep it easily readable
- keep it concise (1 side A4)
- describe the *key steps* in the process
- keep it general (e.g. avoid mentioning code / functions specific to Matlab/LabVIEW)

Developing flow charts is an important skill. It allows you to breakdown, and documents, **how to solve a large complex problem** by performing a **series of logical, small steps**. You (should!) use this approach throughout your degree. *Both LabVIEW and MATLAB coursework assignments in MECH1010 will require the submission of a flowchart.*

CHANGELOG

Version	Changes
Revision 01 – 27.03.2023	First release – basic information
Revision 02 – 01.04.2023	Pin connections for left and right motors (P3) Angle calibration information (P3) Controller speed reduced to 25Hz (P4) Template code and Simulation updated
Revision 03 - 05.04.2023	Updated controller min and max limits (P3 Table 1.)