MODULAR PROGRAMME

ASSESSMENT SPECIFICATION

Module Details

Module Code UFEEHE-30-1	Run 07SEP/1 AY	Module Title Programming in C
Module Leader Ian Johnson	**Module Tutors** Ian Johnson, John Counsell	
Component and Element Number B2	**Weighting: (% of the Module's assessment)** 35%	
Element Description Practical Coursework 2	**Total Assignment time** 21 hours	

Dates

Date Issued to Students 20/02/2008	Date to be Returned to Students 15th May 2008
Submission Place **THE POST BOXES IN N BLOCK FOYERS** **Boxes are open two weeks before submission date**	**Submission Date** 17th April 2008
	Submission Time **2.00 pm**

Deliverables

As per attached specification

Module Leader Signature

Ian Johnson

Second Coursework - February 2008

Introduction

This term you will continue working in the "marco" laboratory. The overall objective for this assignment is to create a control system for a tethered robot that implements a set of behaviours. Whilst this may seem very challenging, in essence it is a continuation of the work you have already done developing the duckshoot game. In that case, you were only required to read a single digital port (the switches) and write to a single digital port (the LEDs). Here you have a much larger number of devices to control.

Overall, a **perfect** completely functional system will gain you 70% without your report. It is not expected that many people will achieve this! You will be working **in pairs or individually** for this assignment.

This assignment is extremely similar to one that has been run successfully by E. Malliris for engineering students both in the past and currently.

This is a large assignment, you will need to work throughout the term to achieve a good mark!

Marco Programming Assignment

Your assignment, due at 2pm on the day as specified on the cover, is to design and implement solutions to the following tasks, as detailed in this document.

Deliverables:

1. A report, detailing your approach, containing your design, source code, testing etc. This **MUST** be spiral bound, and **MUST** contain a section entitled "conclusions" or "review" describing what you have learnt from the laboratory sessions this term. } 30%

2. A linux machine readable, capable of compilation copy of your work. This must be given to your lab tutor at the time of the demonstration, and contain the actual code which will be demonstrated. This should be on floppy disk or cd-rom. The source code will be tested for similarity to the work of others by automated plagiarism detection software, and by submitting this work you consent to this process.

3. A timesheet (see appendix a) **MUST** be included as an appendix to your report. Your lab tutor should be asked to sign this off at every tutorial. **A copy should be retained as evidence of time management for the GDP**

4. A plan (see appendix b) MUST be included. **This should be signed off by your lab tutor before you begin your work.** . **A copy should be retained as evidence of time management for the GDP**

5. A summary of no more than two sides of A4, including relevant URL's or references of what you have discovered from the world wide wide or other sources about subsumption architectures, augmented finite state machines, and Rodney Brooks work. This **MUST** include references. Note: biographical information is not required or relevant.

6. The presentation/viva. The laboratory sessions during the weeks beginning 21/04/08 and 28/04/08, will be given over to demonstrations of your work. You must supply your source code on a floppy disk, from where it will be compiled and tested. **Attendance at the presentation is mandatory for all students**. Your lab tutor will keep the disk. The source code should be **identical** to that supplied in the report. } 70%

WARNING: The University has strict rules on pretending other peoples work is your own, or in sharing work between groups. Having electronic copies of your source code to compare, makes enforcing these rules much easier!

Hints and Notes

The biggest hint is use your PAL sessions!

In order to complete this assignment, you will need to build on your term 1 work. Digital input and output is exactly the same, although you will have more inputs to consider.

Analogue I-O can be performed straightforwardly; lets look at input first:

```
ret=comedi_data_read(device,subdevice,channel,range,aref,&data);
if(ret<0)
   {
     comedi_perror(filename);
     exit(0);
   }
```

Device is returned by device open we met in term one.
Subdevice 0 offers 16 channels of input (subdevice 1 offers 2 channels of output)
Channels are numbered 0..n
Range can be 0 (0v-10v) or 1 (-10v to +10v)
Aref needs to AREF_GROUND
&data is the address of where you want the data to go.

After calling *comedi_data_read* a value between 0 and 4095 inclusive will be placed in data (question: how many bits resolution is that?) or a negative value will be returned to show an error.

Output:

```
ret=comedi_data_write(device,subdevice,channel,range,aref,data);
if(ret<0)
   {
     comedi_perror(filename);
     exit(0);
   }
```

Note: The data this time is actual data **NOT** a pointer to it.

If you want to demonstrate your ability, feel free to use any of the better comedi functions. These are all documented in the online documentation which is available on every machine in the marco lab. What has been presented here is the most straightforward approach.

Programming Tasks

1   Create a menu to enable selection of each of the tasks to be completed by MARCO. The tasks required are shown as items 2 to 6.

2   Allow MARCO to be driven from a given start position using the **analogue** joystick and varying control of the DC motors. In other words, if the joystick is pushed half forward, the motors should drive forward at half speed. The operating mode for the analogue joystick is given below on the left and the expected motor drive on the right. Control of MARCO by the joystick should terminate automatically when MARCO detects the white line.

Joystick position	left motor	right motor
fwd	full fwd	full fwd
fwd right	full fwd	stop
right pivot	full fwd	full back
back right	full back	stop
back	full back	full back
back left	stop	full back
left pivot	full back	full fwd
fwd left	stop	full fwd
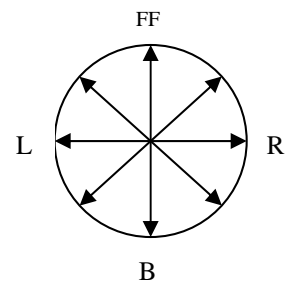
Your program should provide a **smooth and proportional transition** between these control points.

3   Follow the white line using proportional control of the DC motors. When an obstruction is detected by the bumpers MARCO should stop.

4   Move towards the brightest light source (could be more than one and may be moving) whilst continuing to scan for light (scanning **whilst** moving). Your algorithm should cause MARCO to move towards the light source quickly in response to large tracking errors and slowly when the tracking error is small (a proportional response would be ideal). The function should terminate when bumpers detect a collision. A record of the path taken towards the light should be recorded and stored to disk as a data (ASCII) file.

5   Using the data file re-trace the path (recorded in 4) from the starting position to the point where it stopped (approximately).

6   A Behaviour- Based AI Mobile Robotics Exercise. (See Appendix C)

Your robot should follow a white line, and if it loses the line perform a random search to attempt to regain it. If this fails it should look for a bright light, and if one or more are found head towards the brightest. If it encounters an obstacle, the robot should stop and return control to the operator via the analogue joystick.

Appendix A – Sample Timesheet

Date	Start Time	Finish Time	Who Present	Tasks done

Appendix B – An outline plan

You should plan your project and identify two milestones to assist you in achieving the required time-scale. Part of this initial planning should be to divide the work required between the two members of the group. This division of work should be detailed below. The milestones and division of work should be presented to the Module Leader for signature **before** any coding or further planning is started.

Milestones:

	Description (include number of tasks you can realistically accomplish before target date, eg. milestone 1: menu, joystick control , follow light)	Target Date (week beginning)
1		17.03.2003
2		

Division of Work:

Student 1: Name: _____

Allocated Tasks (include here the tasks student 1 will try to accomplish)

Student 2: Name: _____

Allocated Tasks (include here the tasks student 2 will try to accomplish)

Lab tutor (acceptance of milestones and division of work):

Name :_____ Signature: _____

Appendix C
A Behavior-Based AI Mobile Robotics Experiment

1. Background - The Demise of Classical AI for Mobile Robotics

The Behaviour Based Robotics "crusade" that swept through the Artificial Intelligence (AI) research community in the late 1980s and early 1990s started a revolution that was just waiting to happen. By the late 1980s, classical AI approaches had established a very successful reputation in certain important niche application areas. However, in these times, an important focus on "situated" AI emerged, i.e., Artificially Intelligent creatures moving and interacting with their environment. It quickly became clear that the usefulness of the Classical AI approaches in fast reactive, and other low-level, behaviour commonly found in these robotics-dominated areas, was *very* limited. The classical approach to the embedded tasks of goal achievement for a mobile robot in a sensor information-rich environment, seemed to be computationally intractable. The classical, and ultimately unsuccessful, approach involved three phases. Firstly, collect *all* available sensory information (the "sense" phase). Secondly, process *all* this low-level data together (normally in very complex ways) to build an environmental model that can be manipulated in meaningful ways (the "think" phase). Finally, drive the motor interfaces appropriately (the "act" phase). This structure is illustrated in figure 1 below. The first and second phases of this sequential procedure can be *very* computationally demanding, partly because the robot's controller is trying to manipulate *all* of the information centrally.

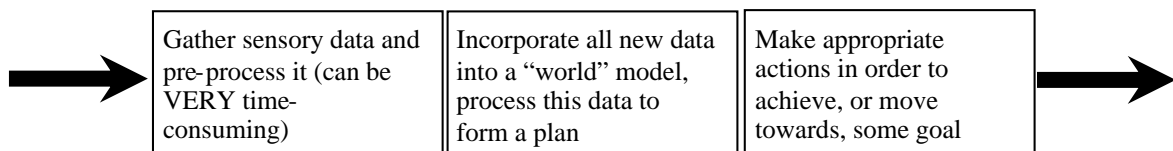| Gather sensory data and pre-process it (can be VERY time-consuming)	Incorporate all new data into a "world" model, process this data to form a plan	Make appropriate actions in order to achieve, or move towards, some goal

Figure 1: The Classical AI method

In addition to all of this, over the 1970s and into the 1980s, there was an increasing realisation that AI had not really made the huge strides forward that had been predicted in the 1950s. Human conscious logical reasoning had been the focus for AI research throughout the 1950s, 60s, 70s and 80s. This *is* still a very important area for research. However, it is only in the areas of long, repetitive and detailed mathematical processing, that computers have *really* made achievements beyond the capabilities of human beings. In fact, in many respects (especially if one thinks widely across the range of successful life on this planet in an evolutionary sense), conscious reasoning is really only the "icing on the cake" of vertebrate (and invertebrate) animals. For example, the majority of the immense number of neurons in the human-brain (10^{10} of them (or 10 billion!), are devoted to the unconscious processes of sensorimotor co-ordination and movement. In other words, staying on our feet, moving and sensing in a dynamic world, is really much harder than sitting and thinking!
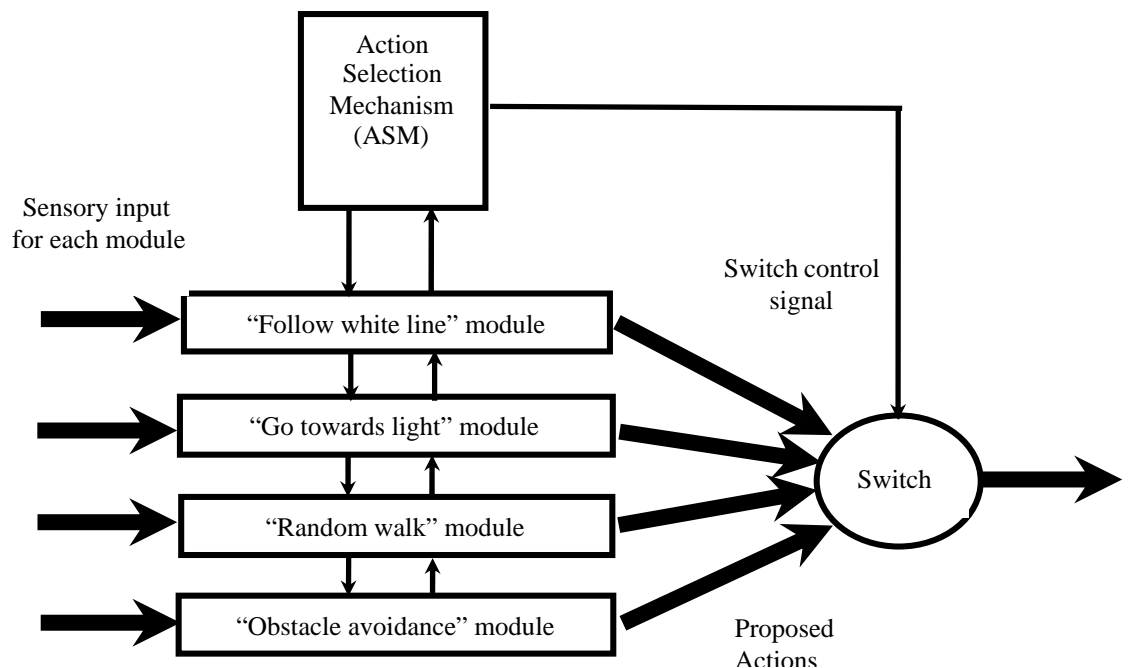
2. The Behaviour Based Approach

Figure 2: The Behaviour Based Approach

It is clear from modern studies of sub- and unconscious processing in the brain and spinal column, that there are many processes going on simultaneously. A US resident Australian researcher, Rodney Brooks, proposed the following simple, but highly successful, model for low-level sensorimotor behaviour. His ideas came at a time when the aforementioned revolution in AI was gaining "steam". The idea was to split-up the information processing of an artificial creature, such as a mobile robot, into small horizontal slices rather than vertically into the three phases described above. See figure 2 above for an example, which might be suitable for use in a mobile robotics application.

In this model, each horizontal slice is a "behavioural module". Each module uses only that sensory data that is relevant to fulfilling its "competency". The way each module processes the sensory data it receives could be complex or very simple, but no matter how complex it is, it is likely to be much more straightforward than the "process everything together" model of classical AI.

Periodically, a behavioural module that has control of the robot's motors will relinquish that control. This could happen because its sensory input is no longer relevant, e.g., a "white line following" module no longer detects a white line.

Alternatively, this might happen because the sensory data means that another behavioural module should take-over, e.g., an "obstacle avoidance" module suddenly needs to act so that an imminent collision is avoided.

In most modern Behaviour Based Architectures an "Action Selection Mechanism", or ASM, mediates between the behavioural modules. When a behavioural module gives up control of the robot, the ASM decides which behavioural module will have control next. Behavioural modules may all have equal "weighting" or priority or, more likely, they may be arranged in a hierarchy.

TASK 6 :   Objective

3. Setting the behaviours you will Implement in an Industrial Context

Let us imagine that an industrial-scale wheeled mobile robot is to be designed to operate in a large mail sorting office. It will tow a set of passive wheeled carts behind it (a bit like a locomotive pulling carriages). On each cart is partially sorted mail. A human member of the sorting staff can stop the robot at any time by putting an obstacle, such as a foot, in the path of the front "bump stops". Whilst stationary, the partially sorted mail on the carts can be loaded and unloaded so that the sorting process can proceed. When the bump stop is released, the robot can continue moving as before.

During normal operation, the robot **follows a white line** laid out on the floor. The white line is movable by the human operators so that they can change the route from time to time. Of course the white line can become scuffed, damaged and obscured over time. If the robot **detects** that the line has disappeared, then it is **to engage** in **"random walk"** behaviour, in search of it. However, if during this "random walk" phase, a central bright light becomes illuminated, then it should act as a "homing beacon" for the robot. Under these circumstances, the robot is **to orient** itself **towards this bright light** in the centre of the sorting office and move towards it. Once the robot has reached this location, it will encounter an **obstacle** that **triggers** its bump stops**.** Whilst this is not an ideal situation, this default location is one in which the sorting process can continue when white line following has gone wrong for a while.

If the robot **re-discovers** the white line, **either** during its "random walking", **or** during its journey towards the light, then "line following" behaviour should re-establish itself. During *any* of this behaviour, if collision with an obstacle is detected, then the robot should **stop moving** immediately, and stay still until that bum stop, or stops, is released.

4. Implementing the Behaviour Based Approach

It is simple to implement a system like that illustrated in figure 2, which achieves the overall behaviour described in section-3 above on the MARCO robot, in stages. Each behavioural module can be designed and debugged as a separate entity, without the ASM structure, i.e., with that behavioural module having permanent control of the robot. This can be been done for each module. Next, each of these modules can be encapsulated within a function.

Following this, a "master" function can be designed, the ASM, that calls each behavioural module and decides what to do when a module finishes for some reason and returns to the ASM. Remember that this could happen either because a module has no further need for control of the robot, or because the sensory information indicates that another module urgently needs control. This can all be implemented using a function "call and return" structure in the C programming language. Let us assume that each behavioural module contains the code for fulfilling its purpose, including the conditions for "giving up" control, which result in returning from the C function that implements it.

The ASM can be the main( ) function or part of a menu system that alternatively allows the human user to activate each behavioural module independently.

An alternative approach is to build a "time slice" scheduler based finite state machine of behaviour modules, although this will require some thinking as to how the scheduler can choose which behaviour should be executed next.

5. In Conclusion

By stages, you can implement your version of some famous experiments from the recent history of AI research by making use of the MARCO robots and their PC-based controllers. This is *fun* to do, and will give you an insight into the important new turns that AI research has taken in recent years. This is still a very important subject that underpins much of the current research being carried out in UWE's internationally renowned Intelligent Autonomous Systems Laboratory in the Du Pont building opposite the main entrance to the University.
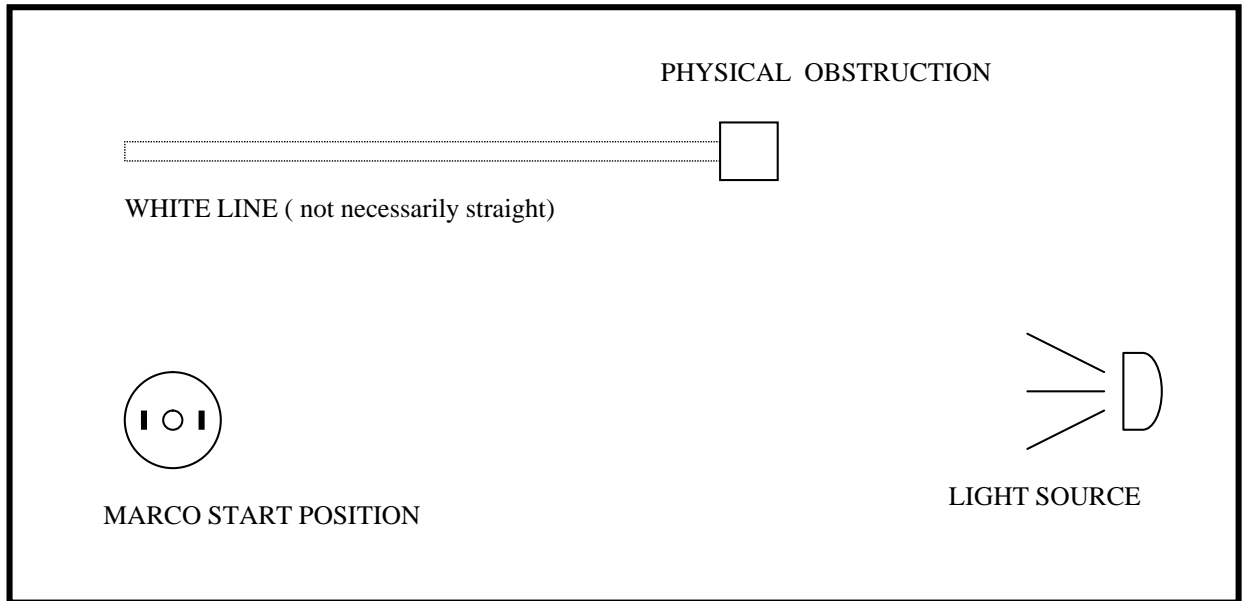
By the way, some of you may notice that you will be doing something in addition to the creation of a Behaviour Based robotics control architecture. You will have begun to construct the elements of a simpler "scheduler" for a basic multitasking operating system. Such schedulers are at the very core of "proper" operating systems like UNIX. We have met the idea of scheduler based finite state machines in the previous terms lectures.

Appendix D   Connecting MARCO

Your project will be assessed by demonstration of your working program. The following connections should be defined in your code as the Rack & MARCO will already be connected in this way.

RACK Digital I/P	Patch Box	*MARCO*	*RACK* Motor Drive	*Patch Box*	*MARCO*
D0	SW1	Left Bumper	P0	SM01	Photodiode stepper
D1	SW2	Right Bumper	P1	SM02	
D2	IRSW1	IR (floor) Switch (left)	P2	SM03	Motor drive and control
D3	IRSW2	IR (floor) Switch (right)	P3	SM04	
D4	SMSW	StepperMotor Switch (left)			
D5	SMSW1	StepperMotor Switch (right)	**DAC Output**		
ADC Input			CH0 (Mtr High)	LDCM+	Left DC Motor Drive
A0	'Y' axis of Analog Joystick		CH0 (Mtr Low)	LDCM-	
A1	'X' axis of Analog Joystick		CH1 (Mtr High)	RDCM+	Right DC Motor Drive
A3	EYE	Photodiode	CH1 (Mtr Low)	RDCM-	

Appendix E   (Possible)Test Track

PHYSICAL OBSTRUCTION

WHITE LINE (not necessarily straight)

MARCO START POSITION

LIGHT SOURCE

Appendix F   Marking Scheme

Demonstration (max. marks: 70)
The following points will be assessed during your demonstration. **None of these marks are available if the demonstration has not been completed. These marks should be allocated and signed off during the demonstration.**

No demonstration   (-10)

User Interface   (4)
- Suitable menu
- No excessive interaction
- Reporting of task in progress
- Reporting at end of task
- Other good features

Joystick Control   (8)
- Calibration Method
- Forward/Back & Left/Right
- Other directions
- Proportional speed
- Control stops when line encountered.
- Records logfile (if not done for light following)

Line Following   (10)
- Follows a straight line
- Follows a line with angles
- Follows a curved line
- Moves at a good speed
- Moves Smoothly
- Stops at an obstruction

Light Following   (18)
- Rotate in correct direction
- Moves towards light at good speed
- Scans during movement
- Stops and scans immediately at an obstacle
- Writes a log file

Replay Path   (10)
- Read log file
- Retrace path to within +/- 15 cm
- No excessive movement
- Tracks timing
- Moves smoothly

50%

Pseudo-subsumption system (AI Execercise)   (20)

- Control module
- Random walk
- Find line
- Stop at collision
- Go to light

20%

Report marking scheme

(Maximum marks 30)

No plan	(-5)
No timesheet	(-5)
Not spiral bound	(-5)

Overall Report Design (layout, presentation, structure, English language)	(5)
Design (system and functions including appropriate diagrams)	(7)
Discussion	(5)
Conclusions/review	(3)
Rodney Brooks report	(5)
Appendix-1 : Program Code Listing (layout, quality of comments, function & program headers etc)	(5)