

Embedded Systems

ENGT2303

Lab Assignment 1

week 5

EQUIPMENT: PC with ReTrO simulation system

EXPERIMENT 1

Build a working CPU with 8-bit data bus (8-bit op-codes, 8-bit operands, 8-bit addresses) and the following ALU/CU functions:

- | | | |
|-------|--|--------------------------|
| • 0 v | load constant | acc := v |
| • 1 v | add constant | acc := acc + v |
| • 2 v | subtract constant | acc := acc - v |
| • 3 a | store accumulator value to memory address a | mem[a] := acc |
| • 4 a | load memory value from address a in accumulator | acc := mem[a] |
| • 5 a | add memory value to accumulator | acc := acc + mem[a] |
| • 6 a | subtract memory value from accumulator | acc := acc - mem[a] |
| • 7 a | branch unconditionally to address pc+a | pc := pc+a |
| • 8 a | branch conditionally if acc = 0 to address pc+a | if acc=0 then pc := pc+a |
| • | all other opcodes are "no operation" | |

EXPERIMENT 2

Write a program to calculate $1 + 2 + 3 \dots + m$, for a given value m. So:

$$result = \sum_{i=1}^m i$$

Data locations: value m in location \$A0
result in location \$A1

Algorithm: clear result_value /* Assume $m \geq 1$ */
loop:
add mem[m] to result
decrement mem[m]
if (m≠0) branch to **loop**

Example:	mem[A0] (m)	mem[A1] (result)
	3	3
	2	5
	1	6
	0	6

Embedded Systems

ENGT2303

Lab Assignment 2

week 6

EQUIPMENT: EyeBot boxes, **Assembly** language

EXPERIMENT 1 "Simon" Game

1. Generate 3 random numbers in the data range [1..4].
2. Play the sound associated for each generated number [1..4]:
AUTone(200,500) *or* AUTone(400,500) *or* AUTone(600,500) *or* AUTone(800,500), resp.
3. Read 3 input button presses, data range [KEY1..KEY4]
For every key press:
 - Convert [KEY1..KEY4] to data range [1..4]
 - Play the associated sound (see above).
 - If the pressed key does matches the generated number, print "OK", else "ERROR".

See flow chart for details and suggested subroutines.

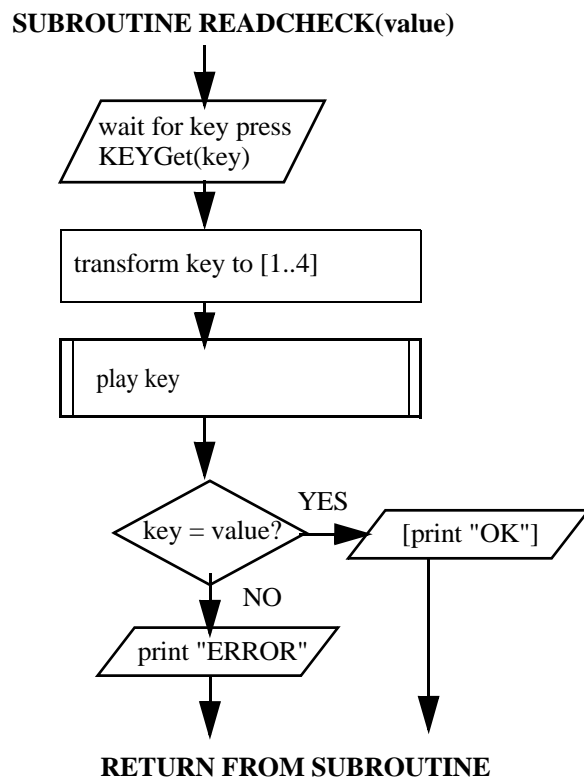
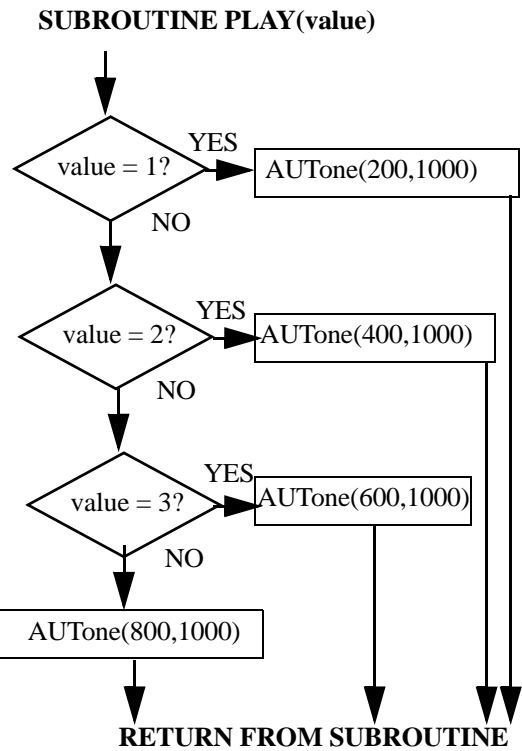
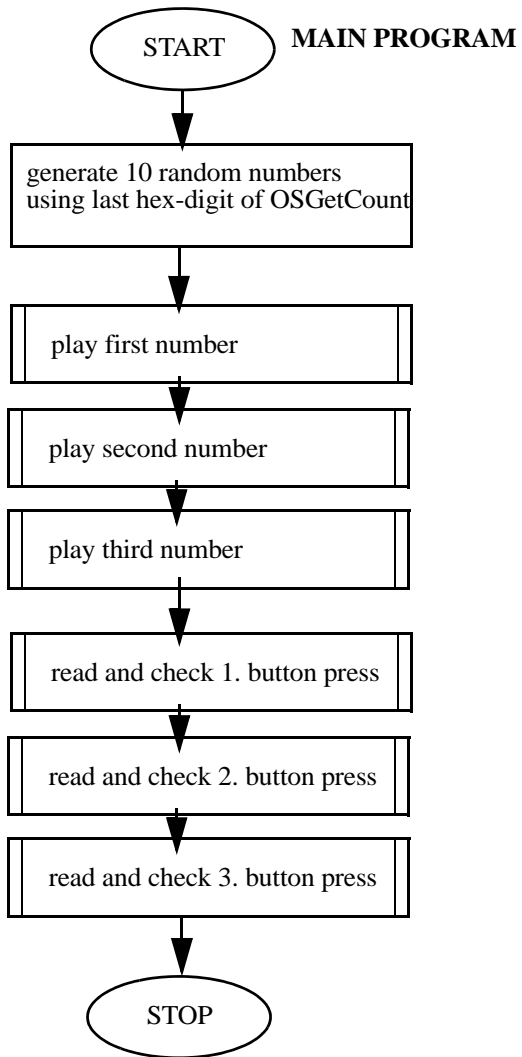
For getting the system time in 1/100 s counts, use the routine:

```
JSR OSGetCount      | returns time since system start in D0
```

For sound output use AUTone:

```
MOVE.L duration, -(SP) | push duration parameter on stack [msec]  
MOVE.L freq, -(SP)    | push tone frequency on stack [Hz]  
JSR  AUTone           | call sound subroutine  
ADD.L #8, SP          | delete two parameters from stack
```

Bonus point: The real senso game starts with a sequence of 1 tone, then for every successful repeat, it increases the sequence length by one tone. Implement this feature for a bonus point.



Embedded Systems

ENGT2303

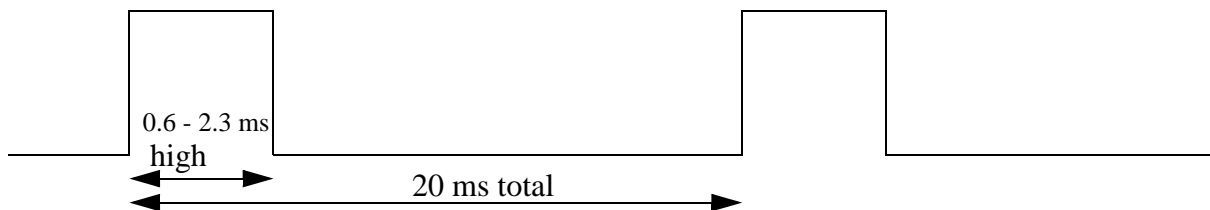
Lab Assignment 3

week 7

EQUIPMENT: EyeBot boxes and interface boxes, **Assembly** language

EXPERIMENT 1 Motor Signal Measurement

Servos are DC motors with built-in logic. They are e.g. used in model cars, airplanes, ships. Servos assume a position (often in the range of 360 degrees) according to an input rectangle signal.



Connect a servo via the **TPU box** (timing processor unit) to the EyeBot. Then use the buttons: Hrd / HDT / Servo / Tst to move the servo to its end positions.

Connect an oscilloscope to the servo's signal pin and analyze the signal.

Write down measurements for signal length and total signal period.

EXPERIMENT 2 Motor Signal Control via Output Latches

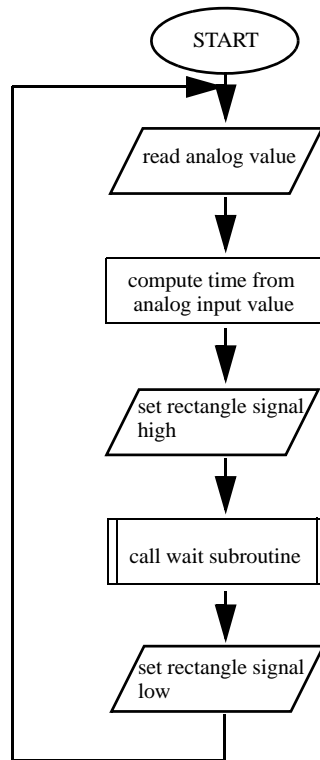
- Connect the servo signal pin to bit 0 of the output port on the EyeBot box.
- Extend the assembly program from the previous experiment to control a servo, setting it to a middle position value.
- In a continuous loop, read key presses and change the servo position towards clockwise (if KEY1 is being pressed) or counterclockwise (if KEY2 is being pressed).
- Print a message to the LCD whenever a button is being pressed, including the new servo position, for debugging purposes.

The servo signal will be connected to bit 0 at the assembly latch address: `OutBase`

Note: - You will need to write a subroutine to wait for variable signal uptimes and downtimes, e.g. 1.5ms uptime and 18.5ms downtime (sum equals 20ms).

- Use the oscilloscope to verify your generated signal **before** you connect the servo!

Program structure:



Bonus point: On the interface box, connect the potentiometer with an analog input of the EyeBot. In an endless loop, read the analog value and print it on the LCD. Set the servo position in accordance to the analog value being read in a continuous loop. For analog input you can use an assembly call to the C RoBIOS function:

```
int OSGetAD(int channel);  
Input:      (channel) desired AD-channel range: 0..15  
Output:    (returncode) 10 bit sampled value  
Semantics:  Captures one single 10bit value from specified  
             AD-channel. The return value is stored in the  
             least significant bits of the 32 bit return value.
```

Embedded Systems

ENGT2303

Lab Assignment 4

week 8

EQUIPMENT: EyeBot boxes and interface boxes, C language

EXPERIMENT 1 Digital Oscilloscope

Write a C program (or a C / assembly combination) to plot an analog input value.

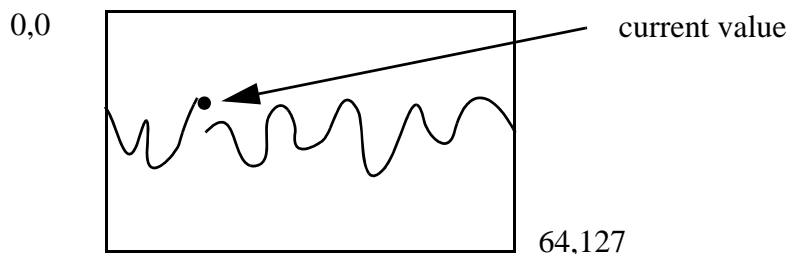
At program start, let the user choose an input channel via a menu button press. These should be: KEY1 = input 0 (microphone), KEY2 = input 2 (connect to potentiometer on interface box), KEY3 = input 4 (connect to potentiometer on interface box), KEY4 = exit.

Make sure to test the minimum and maximum values of the analog input, to use the full screen area for plotting.

Plot the analog signal versus time on the graphics LCD. The dimension of the LCD is 64 rows by 128 columns. For plotting use the function:

```
int LCDSetPixel (int row, int col, int val);
```

Maintain an array of the most recent 128 data values and start plotting data values from the leftmost column (0). When the rightmost column is reached (127), continue at the leftmost column (0) — but be sure to remove the column's old pixel before you plot the new value. This will result in an oscilloscope-like output.



Bonus point:

Plot two analog inputs at once, each using a fraction of the screen height.

Embedded Systems

ENGT2303

Lab Assignment 5

week 9

EQUIPMENT: SoccerBots, C language

EXPERIMENT 1 Calibration of PSD Distance Sensors

1. Perform distance measurements with a PSD sensor for distances 10cm .. 100cm in 10cm intervals. Record the PSD-raw output and draw a graph for raw-output over actual distance.
2. Calculate the regression for the PSD data and fill the PSD lookup-table in the HDT file; upload the new HDT file.
3. Repeat the distance measurement with the final PSD values (corrected with HDT table) and redraw the same graph.

For reading distance values read the EyeBot documentation and use the following routines:

```
PSDHandle PSDInit(DeviceSemantics semantics);  
int PSDRelease(void);  
int PSDStart(PSDHandle bitmask, BOOL cycle);  
int PSDStop(void);  
BOOL PSDCheck(void);  
int PSDGet(PSDHandle handle);
```

EXPERIMENT 2 Wall Following

Implement a program that drives the vehicle along a wall using a single PSD sensor (e.g. left sensor). Use subsequent PSD readings to determine whether the vehicle gets closer to the wall (then steer away), remains at the same distance or gets further away (then steer closer).

- The wall will either be straight or slightly curved, but it will not have a sharp bend. In the laboratory, use the border wall of the experimentation area.
- The robot should update its PSD sensor readings periodically and plot the trajectory it is driving on its LCD.
- Use the robot's front PSD sensor to detect an obstacle (e.g. wall in front) and stop the robot.

For implementing this experiment, you may use the motion sequence from the previous experiment or you may implement a P-controller that lets the robot drive a curve to maintain a constant distance from the wall.

Embedded Systems

ENGT2303

Lab Assignment 6

week 10

EQUIPMENT: SoccerBots, C language

EXPERIMENT 1 “Bang-Bang” Motor Control

Write a motor controller in C implementing Bang-Bang control for a single motor
The program should read a speed value from the input buttons and maintain a constant wheel speed irrespective of changing load.

Follow the algorithm and procedure outlined in the lecture notes!

Display the current encoder value and motor speed on the LCD, using a suitable time interval to see the impulse response of the motor.

Start recording 500 speed values in an array after pressing the button for changing the motor speed. Use another button for transmitting all values as text to the PC via RS232, using the routines `fprintf` or `OSSendRS232`. Make sure to insert a newline after each value.

For receiving on the PC, use the command: `ul filename`.
Use Excel or similar in graph mode for visualizing your results.

EXPERIMENT 2 PID Motor Control

Write a motor controller in C stepwise implementing a PID controller for a single wheel.
The program should read a speed value from the input buttons and maintain a constant wheel speed irrespective of changing load.

Follow the algorithm and procedure outlined in the lecture notes!

Step 2.1

P-Controller (proportional only)

Step 2.2

PD-Controller (add derivative component)

Step 2.1

PID-Controller (add integral component)

Display encoder and speed values on the screen and upload speed values as in Experiment 1.

Embedded Systems

ENGT2303

Lab Assignment 7

week 11

EQUIPMENT: SoccerBots, C language

EXPERIMENT 1 Object Detection

For this experiment, a single **constant color image** will be provided that needs to be included with your application program. The grayscale and color image data types are defined as:

```
typedef BYTE image[imagecolumns][imagerows];  
typedef BYTE colimage[imagecolumns][3];
```

A grayscale image contains 1 data byte per pixel, a color image contains 3 data bytes for each pixel, representing values for red, green, and blue.

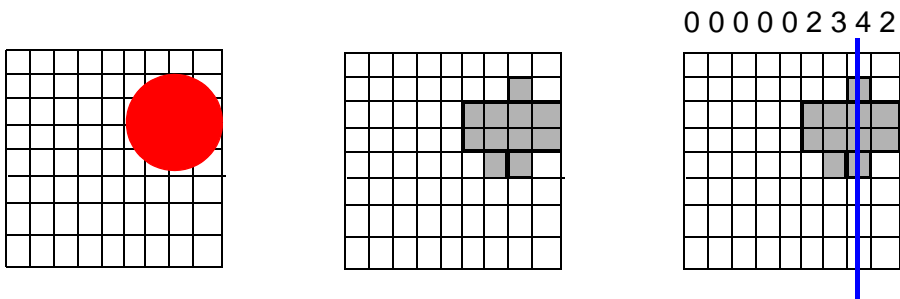
Implement a program to detect a red object in the image.

Step 1: A pixel is considered “red”, if

- its red component is greater than 150,
- its green component is less than 75, and
- its blue component is less than 75.

Write a conversion routine that transforms a given “colimage” into a binary image, for which a pixel value is “1” if the pixel is red, and “0” otherwise.

Display this binary image on the LCD.



Step 2: Write a loop to go through the converted image, maintaining an additional arrays with one entry for each column for counting the number of pixels equal to “1” in that column.

Step 3: After finishing the loop over all pixels, analyze the column array. This is now a **histogram**. Find its maximum value and print it. If this value is above a threshold (e.g. 6) print that an object has been detected. Mark the detected object center in the image, if a detection is made. Otherwise print "none" to the LCD.

EXPERIMENT 2 Object Tracking

Use your solution of the previous experiment to implement a tracking system. Your program needs to detect a red object in a camera image and use the vehicle's camera pan servo for directing the camera straight at the red object.

Read an image from the camera using RoBIOS function:

```
int CAMGetFrame(image *buf);
```

You should be using a P-controller for tracking the object and trying to minimize the difference between the object center's x-coordinate and the center of the image.

Embedded Systems

ENGT2303

Lab Assignment 8

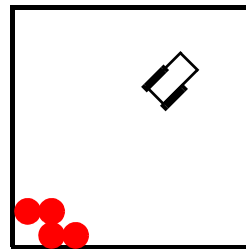
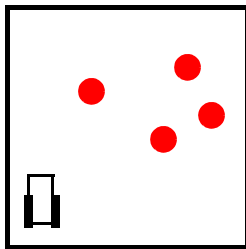
week 12

EQUIPMENT: SoccerBots, C language

EXPERIMENT 1 Collecting Cans

This experiment brings together all your experience from the previous lab sessions.

The vehicle is started in one corner of the experimentation area with a number of red cans placed around the field.



Your task is to:

- Drive the robot inside the experimentation area (“wander”)
- Search for red cans (use camera)
- Avoid bumping into walls while driving (use PSDs)
- When a red can has been located, drive towards it and lock it by switching on the vehicle’s electromagnet
- After locking a can, drive back to the robot’s starting position (“home area”) and release the can there.
- Continue driving and looking for more cans.

Provide a detailed report about this lab assignment before your lab session. This report has to be handed in to your lab supervisor before starting work in your lab session and will be marked as part of this lab.