Group 3: Mark Baker, Melyssa Fowler, Jie Fang, Gary Steelman

Assignment 2

Control a POB-System Robot to Safely Cross a Railway Track

Purpose

The project's main objective is to gain experience designing a real-time system for a real-world robot. The robot has limited capabilities, thus care must be taken to allot time enough for each task the robot must accomplish. In this project the robot is to cross a railway track safely. It must avoid oncoming train engines and other obstacles (ie other robots) as it crosses. Careful consideration of the capabilities of the robot lead to various design decisions elaborated below.

The Robot

The Hardware

The robot used is a POB-System robot consisting of various POB modules. As was supplied for this project, the inputs and outputs for the robot are below.

POB-System Inputs

POB-Eye

- 8-bit color camera with resolution of 88x120 pixels.
- The POB-Eye can classify incoming images containing special symbols.
- Can look up and down.

Joystick

- A simple joystick connected to the motherboard.
- The joystick can be depressed for a digital clicker.
- Can be used for analog input.

Serial port

- A power port on the motherboard allows a connection from a serial port to download a program module.

POB-System Outputs

6 Servos

- Values range from 0 to 255.
- Yield small amounts of torque enough for small amounts of rotation.

2 DC motors

- The DC motors hand current up to 600 milliamps.
- Yield larger amounts of torque than the servos, but consume more power.
- up to 600mA

POB-LCD128

- A simple black and white LCD screen with resolution 128x64 pixels attached to the top of the POB-Bot.
- Bit per pixel and byte per pixel program modes are supported.

In the Context of Railway Crossing

Of all the capabilities the POB-Bot has, only a subset are truly useful in the context of railway crossing:

Useful Capabilities

Using the POB-Eye to rotate up and down for a wider field of view allows the robot to see more, thus increasing its sensitivity to danger. Capturing an image from the continuous camera stream followed by pattern recognition and classification on the image allows the robot to distinguish between various objects. This means the robot can distinguish between trains and other robots. Displaying images on the LCD screen will be useful for testing and debugging the program code. The joystick can be used to switch between various modes in the program code. And most importantly, the servos and DC motors allow the robot to move, thus allowing it to cross the track.

Limitations

The POB-Bot system is not a high-end robotic system. While it is convenient for hobby use, it has many limitations. It cannot move quickly because the power to the servos must be limited for fear of burn-out. The DC motors can only handle 600 milliamps of power. The POB-Eye's camera resolution is very low, which will require the robot to be close to an object before it can recognize the object. The processor in the motherboard has a low clock rate, thus leading to slower program execution than many are accustomed to on a desktop computer. The motherboard also contains a low amount of memory as compared to a desktop computer. Care must be taken to work within these limitations.

Program Design

Assumptions

It is impossible to reach consensus with all groups on this project. Thus we define our own system for recognizing trains and other robots.
Trains will be marked with a special symbol on the front and sides and other POB-Bots will be marked with a special symbol on the front. The symbol for trains and for POB-Bots will be different.
We also define how avoidance of other robots works. Two POB-Bots which detect each other will rotate 90 degrees clockwise, proceed forward a short distance, and may then return to their original heading.

Tasks

The POB-Bot program is decomposed into a set of tasks. Most tasks are periodic, but we include some aperiodic and sporadic tasks. Below is the task chart. Note that numbers are shown in "units" as the units are yet to be defined.:

Task Name	Abbreviation	Period (units)	Execution Time	Relative Deadline	Release Time
Look Left	ll	40	4	37	0
Look Right	lr	40	4	37	0
Look Center	lc	40	1	37	0
Realign Left	rl	40	4	39	4
Realign Right	rr	40	4	39	4
Capture Image Left	cil	40	1	38	4
Capture Image Right	cir	40	1	38	4
Capture Image Center	cic	40	1	38	4
Decipher Image Left	dil	40	1	39	5
Decipher Image Right	dir	40	1	39	5
Decipher Image Center	dic	40	1	39	5
Initiate Crossing Task	ict	40	1	40	6
Sporadic					
Cross Track			4	20	
Aperiodic					
Avoid Robot			10	80	
Approach Track			5	80	

The tasks are described in detail below. Note that left and right are defined with respect to the robot's perspective.

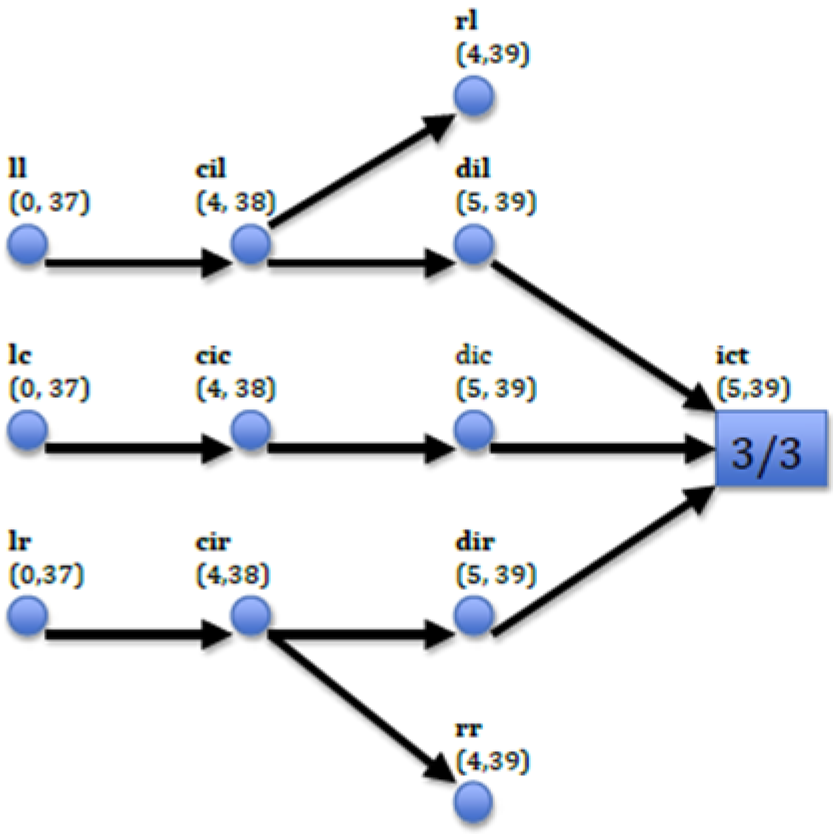
- Look Left

  - The robot rotates to the left an experimentally determined angle less than 90 degrees and prepares to look for a train.
- Look Right
  - The robot rotates to the right an experimentally determined angle less than 90 degrees and prepares to look for a train.
- Look Center
  - The robot prepares to look for a robot straight ahead.
- Realign Left
  - The robot rotates to face perpendicular to the track from facing left.
- Realign Right
  - The robot rotates to face perpendicular to the track from facing right.
- Capture Image Left
  - The camera triggers and saves a picture into the robot’s memory while facing left.
- Capture Image Right
  - The camera triggers and saves a picture into the robot’s memory while facing right.
- Capture Image Center
  - The camera triggers and saves a picture into the robot’s memory while facing center.
- Decipher Image Left
  - The robot processes the saved image to determine if there is a train approaching from the left.
- Decipher Image Right
  - The robot processes the saved image to determine if there is a train approaching from the right.
- Decipher Image Center
  - The robot processes the saved image to determine if there is a robot across the track from it.
- Initiate Crossing Task
  - The robot will do one of three things
    - If there is a robot across from it, it will trigger the Avoid Robot task.
    - If there is not a robot across from it and there is a train approaching, it will do nothing.
    - If it is safe to cross, it will initiate the Cross Track task.
- Cross Track
  - The robot moves forward across the track.
  - This task is sporadic and will always be schedulable with the given schedule.
- Avoid Robot

-
    - The robot rotates 90 degrees to the right, moves forward and then rotates back 90 degrees to the left.
    - This task is aperiodic.
- Approach Track
    - The robot moves forward until it is close to the track but not in danger of being hit.
    - This task is aperiodic, and will only be executed at program startup one time.

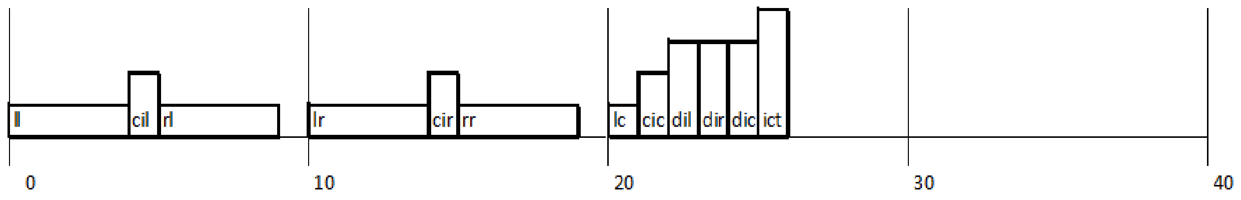
Task Graph

In the figure below, task names are shown followed by their release times and deadline in parentheses. Only the periodic tasks are shown. The *look left, look right,* and *look center* tasks are not dependent on any other tasks. The *capture* tasks are dependent on the *look* tasks because an image cannot be captured unless the robot is looking in the right direction. Following the *capture* tasks, the *decipher* tasks can start deciphering the image looking for trains or robots. Following the *capture image left* and *capture image right* tasks the camera must always realign to center; this is why the *realign* tasks are dependent on the completion of the *capture* tasks. Finally the *initiate crossing task* cannot make a decision on whether to cross or not unless all three *decipher* tasks have completed. Although not shown in the task graph above, the sporadic task *cross track* and the aperiodic task *avoid robot* may be released by the *initiate crossing task.* Furthermore, all of the *look* tasks do not begin until the aperiodic task *approach track* has completed. Another constraint not denoted on the diagram above is that after a *look* task begins, no other *look* task can begin until the correct *capture image* task has completed and for *look right* and *look left* the corresponding *realign* task must also complete. This is to avoid missing the capture of one direction properly.



Scheduling Method

The schedule used is a cyclical executive since it allowed us to schedule such that no Look, Capture Image, Realign task segments overlapped with other directions, ie Look Left, Look Right, Capture Image Left. This would cause inaccurate results. The frame size is 10 as we defined our period to be 40 and our longest task is Avoid Robot at 10, which evenly divides 40. The minimum relative deadline is 20 which means that there is always at least 1 frame between the release time and deadline of every job.



Testing

After implementation the program behavior must be verified. To accomplish this, we will use a piece

of paper folded into a box with the train symbol glued onto it and another piece of paper folded into a box with the POB-Bot symbol glued onto it. This gives us control over the testing procedure and minimizes risk of damage to the robot, other robots, and the train system.
We will test many cases including:

- Train or other robot is far away.
- Train or other robot is at oblique angle.
- Train or other robot is straight ahead.
- Train or other robot is close.
- Train and robot in view at the same time.
- Train or other robot come into view during each combination of Capture Image tasks.
- Robot passes all 3 Decipher tasks.
- Robot fails at least one of the Decipher tasks.

We hope by testing these cases the system produced will safely guide the robot to the other side of the tracks, where in fact, the grass will be greener.

References

http://www.robotshop.com/ca/content/PDF/pob-avoider-manual.pdf