

## ENGR 101: Robotics Lecture 4 – Making Decisions

- Outline
  - The Stall Sensor
  - Making Decisions
  - Random Number Generation
- References
  - <http://cssserver.evansville.edu/~richardson/>
    - PBASIC Programming Guide: Setting Up
    - PBASIC Programming Guide: Writing Programs
    - BASIC Stamp Syntax and Reference Manual

1

## Lecture 4 – Making Decisions The Stall Sensor

- We have used pins only for output, but it is also possible to use a pin for input.
- Pin 7 is connected to a *stall sensor*. When both motors are on and the wheels are turning the sensor sends a low voltage to pin 7. If the motors are on and BOTH wheels are stuck (not turning) the sensor sends a high voltage to pin 7. A low voltage is interpreted as a binary 0, while a high voltage is a binary 1.

2

## Lecture 4 – Making Decisions The Stall Sensor

- We can refer to the value at pin 7 using the PBASIC bit name for the pin, IN7:

```
LOW RtMotor      'Motor init.
LOW LtMotor
PAUSE 100
PULSOUT RtMotor, 2400 'Start your
PULSOUT LtMotor, 2400 'engines...
DO
  DEBUG HOME, DEC IN7
  PAUSE 50
LOOP
```

3

## Lecture 4 – Making Decisions The Stall Sensor

- This program continuously displays the state of the stall sensor (0 – running, 1 – stalled) in a loop.
- It is preferable to label the stall sensor pin:

```
Stall  PIN 7
DO
  DEBUG HOME, DEC Stall
  PAUSE 50
LOOP
```

4

## Lecture 4 – Making Decisions The Stall Sensor

- We will typically want to “read” the stall sensor setting at the beginning of a loop and then do something depending on the setting. You may want to store the value in a variable:

```
Stall  PIN 7
stuck VAR Bit 'Declare var
DO
  stuck = Stall
  'WE ARE STUCK - DO SOMETHING!!!
LOOP
```

5

## Lecture 4 – Making Decisions IF-THEN-ELSE-ENDIF

- Robots (and computer programs) seem intelligent because they can respond differently to different events. The standard computer program decision structure is an IF-THEN-ELSE-ENDIF block:

```
IF (stuck = 1) THEN
  'Code to stop motors goes here
ELSE
  'Code for full ahead goes here
ENDIF
```

6

## Lecture 4 – Making Decisions IF-THEN-ELSE-ENDIF

- If the *condition* is TRUE the THEN block of code is executed. If FALSE the code in the ELSE block is executed. The parentheses around the *condition* are optional but recommended.
- The comparison operators that can be used in a condition are: =, <>, >, <, >=, <=. The operators compare for equality, inequality, greater than, less than, greater than or equal, and less than or equal respectively.

7

## Lecture 4 – Making Decisions IF-THEN-ELSE-ENDIF

- Condition negation uses the NOT operator. Compound conditions can be constructed using AND, OR, and XOR operators:  

```
IF ((stall=0)AND(count<=10)) THEN
  PULSOUT RtMotor, 2400
  PULSOUT LtMotor, 2400
ENDIF
```
- Note that an ELSE code block is optional and that multiple statements are permitted in a THEN (or ELSE) block.

8

## Lecture 4 – Making Decisions IF-THEN-ELSE-ENDIF

- If the Scribbler gets stuck, you will normally want to take some action. This code shuts the motors down for two seconds:

```
IF (stuck = 1) THEN
  PULSOUT RtMotor, 2000
  PULSOUT LtMotor, 2000
  FREQOUT SPEAKER, 500, 440
  PAUSE 1500
ELSE
  PULSOUT RtMotor, 2600
  PULSOUT LtMotor, 2600
ENDIF
```

9

## Lecture 4 – Making Decisions Random Behavior

- To program random behavior, we need a *random number generator*. There is one built into the Scribbler:

```
maxval CON 20 '# 0 to 19
seed CON $AAAA 'RNG seed
dice VAR Word
dice = seed
DO
  RANDOM dice
  DEBUG DEC dice//maxval, CR
  PAUSE 500
LOOP
```

10

## Lecture 4 – Making Decisions Random Behavior

- To use the RNG we must first seed the RNG variable. The RANDOM command will generate a new random number between 0 and 65,535 (since dice is word sized).
- A // is the remainder (or modulus operator), dice//20 will always return a number between 0 and 19.

11

## Lecture 4 – Making Decisions Assignment

- Program the Scribbler so that when it becomes stuck, it should back up a short, random distance, turn through a random angle and then proceed forward. When not stuck it should go forward in a straight line.
- Add LED flashes and sounds as desired ...

12