Exploring Robotics with DREAM system

Chi N. Thai

Topics

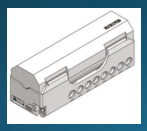
- Description of DREAM robotic system: hardware and software.
- Using the TASK tool to do Basic Programming of Continuous Turn motors and integrated/discrete IR Distance sensors.
- Foundations of Computer Programming – Sequences, Loops and Conditionals.
- Problem Solving using Sense-Think-Act paradigm.
- Programming Avoider's basic autonomous behaviors.
- Remote control of robot using the Virtual Remote controller.
- Modification of Avoider into a Line Follower bot (hardware modification and programming approach).
- Using 3rd IR Sensor to Avoider/Line Follower bot to allow it to scan/detect an object and then to approach the object up to a given distance and stop, and perhaps swing it around and then to follow the track in the other direction.

ROBOTIS DREAM system – Set A

ROBOTIS DREAM System – Set A (L.1)

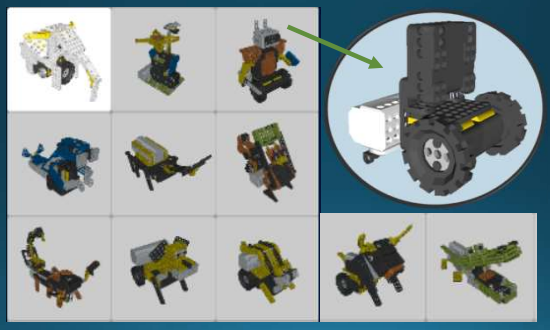
- 13 robot designs based on linkages and gears.

Micro USB Cable to connect to PC for charging battery



ROBOTIS DREAM System – Set A (L.2)

- Details of DREAM Level 2 (11 robots):



ROBOTIS Rivet System

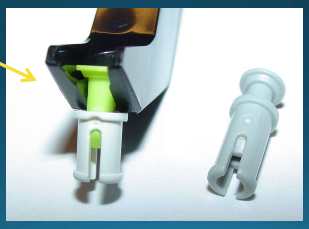
- Rivet components: Sleeve and Stem

Locked position

- Two sizes:
  - Long (grey only)
  - Short (multi-color)

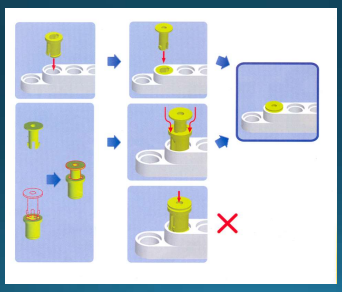
Rivet Tool

Fingers End



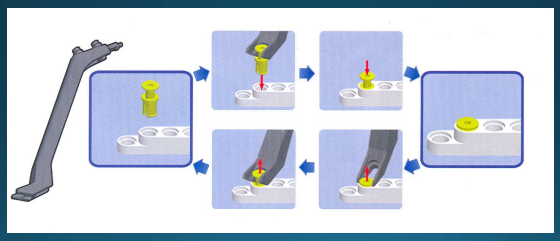
Rivet Attachment Method (1)

- Insert SLEEVE first, then STEM.

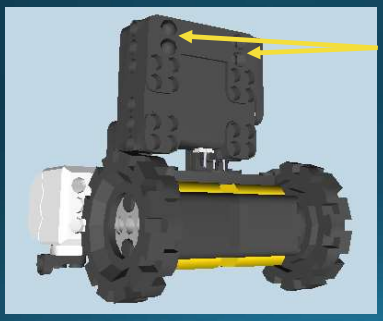


Rivet Attachment Method (2)

- Rivet Pre-assembled + Rivet Tool



Avoider Bot (Pre-built)



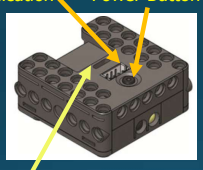
Integrated IR sensors (short range)

DREAM system's Brain

- Controller CM-150

Communication Port

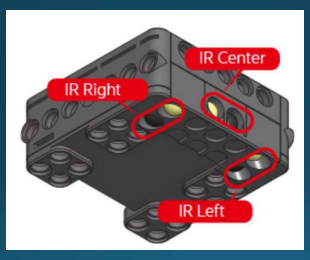
Power Button



Micro USB Cable to connect to PC for Robot Programming (L1) & to charge battery

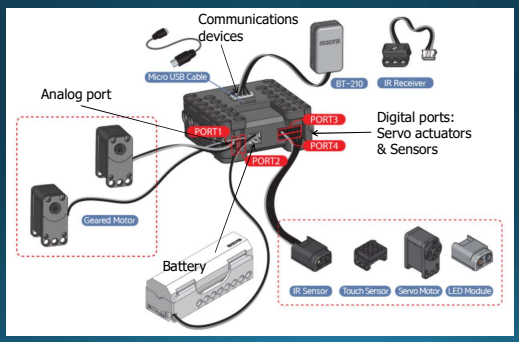
Details of CM-150 (1)

- Controller block & built-in NIR sensors.



Details of CM-150 (2)

- Controller block & attachments.



Microcontroller Concepts

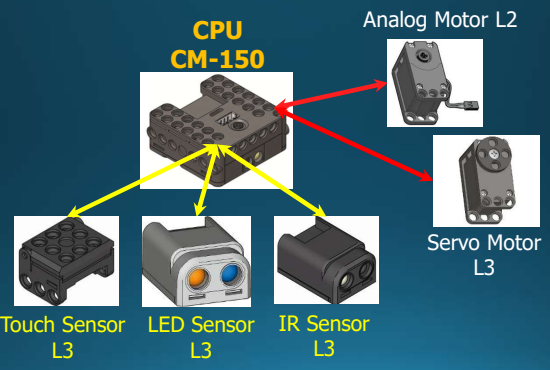
- Controller Chip – STM32F103C8 (72 MHz clock rate):
  - Single-cycle multiplication and hardware division.
- Memory (Flash 64 KB).
- CM-150 with built-in NIR and Sound sensors:
  - Pre-defined procedures (ROM) – Control Table.
  - Parameters values (R/W).

Computer Language

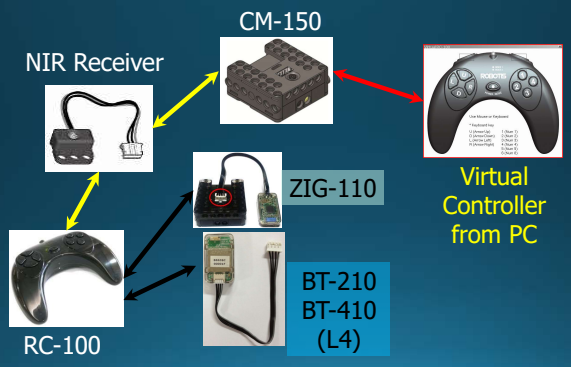
- DECIMAL (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
- BINARY (0 or 1) - Hexadecimal (0-9, A-F).
- '1' or '0' digit is called Bit.
- Byte (8 bits).
- Binary MEMORY Address and Value.
- 8 bit data (0-255).
- 10 bit data (0-1023).
- 16 bit data – High Byte and Low Byte.

0110011111100110

Controller, Actuators and Sensors

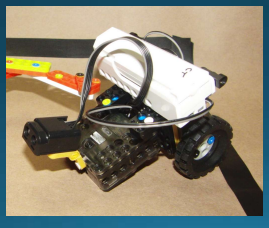


Remote Control Options

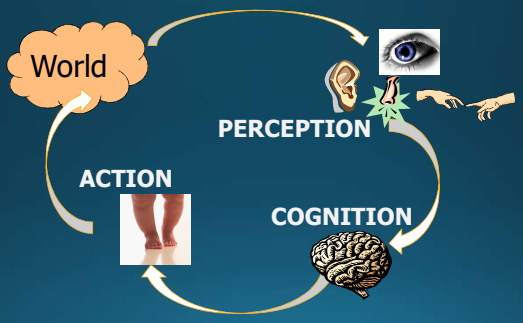


Programming

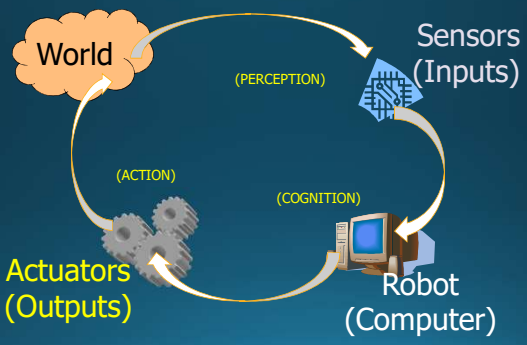
- A robot cannot do anything unless it has received a set of instructions called a program that tells it what to do, how and when to do it.



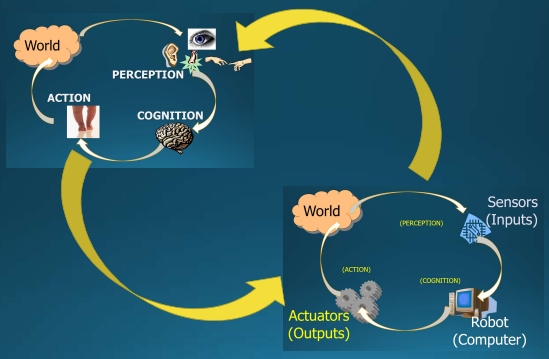
World – Human Interactions



World – Robot Interactions



Robotics Problem Solving Approach



Reactive Control Approach

Given Condition >> Appropriate Robot Action

Conditions	Actions
Condition 1	Action A
Condition 2	Action B
Condition 3	Action C
Condition 4	Action B

- One and Only One Condition happens at any one time
- Multiple Conditions can happen at any one time

Input-Output Table

Given Input Sensor(s) >> Activate Appro. Actuator(s)

Input Sensors	Output Actuators
NIR	DC MOTORS
TOUCH	SERVO MOTORS
MICROPHONE	BUZZER-SPEAKER
LIGHT LEVEL	LEDs

DREAM Programming Tools

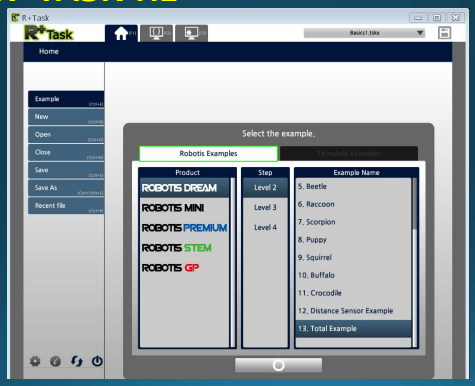
CM-150

Micro USB cable

```
START PROGRAM
{
  Speed = 500
  Print Screen with Line = Speed
  PORT[1]:Geared Motor = CCW:0 (0.00%) + Speed
  PORT[2]:Geared Motor = CW:0 (0.00%) + Speed
  High-resolution Timer = 1,000sec
  WAIT WHILE ( High-resolution Timer > 0 )
}

FUNCTION Forward
{
  PORT[1]:Geared Motor = CCW:0 (0.00%) + Speed
  PORT[2]:Geared Motor = CW:0 (0.00%) + Speed
  High-resolution Timer = 1,000sec
  WAIT WHILE ( High-resolution Timer > 0 )
}
```

R+TASK V. 2

R+TASK V.2



R+TASK V.2

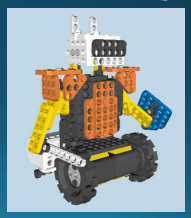
- Design main program on the CM-150 that communicates to Actuators & Sensors (*.tskx).

Programming Action Sequences

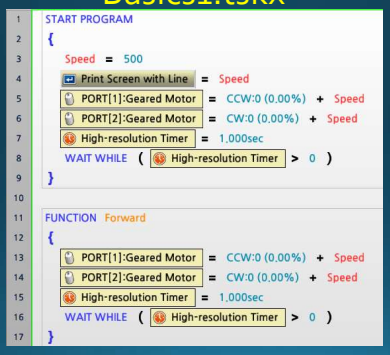
Conditions	Actions
~~Condition 1~~	Action A
~~Condition 2~~	Action B
~~Condition 3~~	Action C
~~Condition 4~~	Action B

R+TASK Programmer Basic Programming (1)

- Print & Println procedures.
- How to turn on/off the Geared Motors.
- How to use the general-purpose Hi-Res Timer.
- How to make Bugbot go Forward, Backward, Left & Right for given time periods.
- Saving work done as Functions.

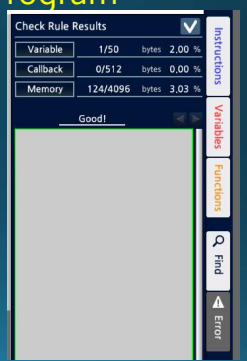


Basics1.tskx



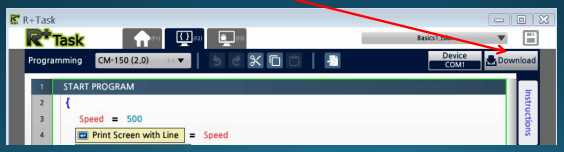
Check the Program

- Click on the "Error" tab and to check the correctness of your code.
- It will show you any error(s) and the line number(s) where errors occurred.
- Fix any errors until you get "Good!" (i.e. no error)



Download the Program

- Attach the USB cable to the CPU.
- Turn on the CPU.
- Choose COM port.
- Click "Download" on right corner of Menu Bar.



Basics2.tskx (sequence of motions)

```
START PROGRAM
{
  Speed = 500
  Print Screen with Line = Speed
  CALL Forward
  CALL LeftTurn
  CALL RightTurn
  CALL Stop
  CALL Backward
  CALL Forward
  CALL Stop
}

FUNCTION Forward
{
  PORT[1]:Geared Motor = CCW:0 (0.00%) + Speed
  PORT[2]:Geared Motor = CW:0 (0.00%) + Speed
  High-resolution Timer = 1.000sec
  WAIT WHILE ( High-resolution Timer > 0 )
}
```

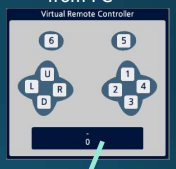

Basics3.tskx (Motion with its own delay)

```
START PROGRAM
{
  Speed = 500
  Print Screen with Line = Speed

  CALL Forward
  TimeDelay = 1000
  CALL Delay

  CALL LeftTurn
  TimeDelay = 250
  CALL Delay

  CALL RightTurn
  TimeDelay = 350
  CALL Delay
```

```
FUNCTION Delay
{
  High-resolution Timer = TimeDelay
  WAIT WHILE ( High-resolution Timer > 0
}
```

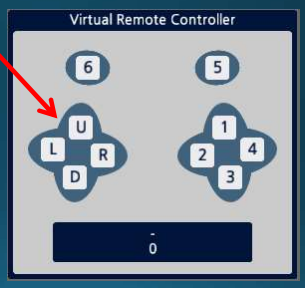

Remote Control of Avoider

RC-100 Virtual Controller from PC



Run the Program with Virtual Controller

- Use the virtual controller to move the bot around
- In Robotics 2, we'll use the physical RC-100 also.



Reactive Control Approach

Given Condition >> Appropriate Robot Action

Conditions	Actions
Condition 1	Action A
Condition 2	Action B
Condition 3	Action C
Condition 4	Action B

- One and Only One Condition happens at any one time
- Multiple Conditions can happen at any one time

Input-Output Table

Given Input Sensor(s) >> Activate Appro. Actuator(s)

Input Sensors	Output Actuators
NIR	DC MOTORS
TOUCH	SERVO MOTORS
MICROPHONE	BUZZER-SPEAKER
LIGHT LEVEL	LEDs

RC-Basics1.tskx

```
START PROGRAM
{
  Speed = 500
ENDLESS LOOP
{
  WAIT WHILE ( Remocon Data Received == FALSE (0) )
  Button = Remocon RXD

  IF ( Button == U )
  {
    CALL Forward
  }
  ELSE IF ( Button == D )
  {
    CALL Backward
  }
```

IF-ELSE-IF for 1 condition at a time

RC-Basics2.tskx

```
// Robot stops when NO button was pushed

ELSE IF ( Button == -- )
{
  CALL Stop
}
```

RC-Basics3.tskx

```
// Robot stops when a button OTHER than U-D-L-R is pushed or NO button was pushed

  ELSE IF ( Button == R )
  {
    CALL RightTurn
  }
  // ELSE IF ( Button == -- )
  ELSE
  {
    CALL Stop
  }
```

RC-Basics4a.tskx

```
Data = Remocon RXD
U_Button = Data & U
D_Button = Data & D
L_Button = Data & L
R_Button = Data & R
IF ( Data == -- )
{
  CALL Stop
}
IF ( U_Button > 0 )
{
  CALL Forward
}
IF ( D_Button > 0 )
{
  CALL Backward
}
```

```
// Robot accepts multiple buttons
// But last command in the list wins
```

Parallel IFs used for possible multiple conditions at once but last command in the list "wins".

RC-Basics4b.tskx

```
IF ( U_Button > 0 )
{
  CALL Forward
  CALL Delay
}
IF ( D_Button > 0 )
{
  CALL Backward
  CALL Delay
}
```

Parallel IFs used for possible multiple conditions at once and with individual time delays.

RC-Basics4c.tskx (Combo approach)

```
IF ( Data == -- )
{
  CALL Stop
}
ELSE
{
  IF ( U_Button > 0 )
  {
    CALL Forward
    CALL Delay
  }
  IF ( D_Button > 0 )
  {
    CALL Backward
    CALL Delay
  }
}
```

User releases any button

User presses U-D-L-R button(s)

Programming Challenge 1

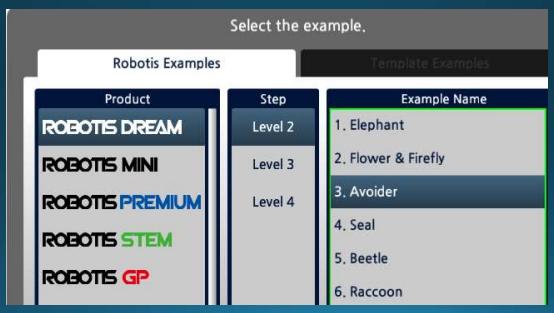
- BUTTON 1 >> Low Speed (Speed = 300)
- BUTTON 4 >> High Speed (Speed = 700)

R+TASK Basic Programming (2)

- How to use built-in musical melodies.
- How to count sound claps.
- How to read Integrated IR sensors:
  - Left, Center & Right.

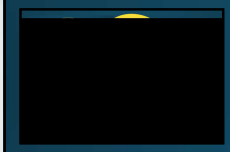
Autonomous Avoider (1)

- Dream_l2_Avoider.tskx



Reactive Control Approach

Given Condition >> Appropriate Robot Action



Conditions	Actions
Condition 1	Action A
Condition 2	Action B
Condition 3	Action C
Condition 4	Action B

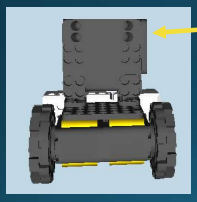
- One and Only One Condition happens at any one time
- Multiple Conditions can happen at any one time

Autonomous Avoider (2)

```
32  }
33  ELSE IF ( IR Left >= 100 && IR Right >= 100 )
34  {
35    // When both sensors detect an object, robot moves backward and turns right.
36    PORT[1]:Geared Motor = CW:0 (0.00%) + Speed
37    PORT[2]:Geared Motor = CCW:0 (0.00%) + Speed
38    CALL Standby_0_5sec
39    PORT[1]:Geared Motor = CCW:0 (0.00%) + 0
40    PORT[2]:Geared Motor = CCW:0 (0.00%) + Speed
41    CALL Standby_0_25sec
42  }
43  ELSE
44  {
45    // Moves forward for all other cases.
46    PORT[1]:Geared Motor = CCW:0 (0.00%) + Speed
47    PORT[2]:Geared Motor = CW:0 (0.00%) + Speed
48  }
```

Challenge 2 –Hardware & Software

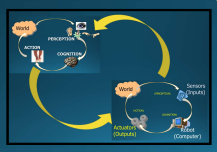
Modify Avoider so that it can follow a black curved track



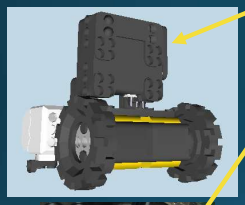
IR sensors
Avoider's eyes

Challenge 2 –Hardware & Software

What are the appropriate threshold values for IR Left & Right sensors to use to describe "off-track" and "on-track" conditions?



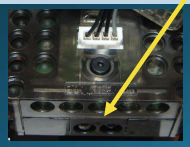
Challenge 3 –Hardware & Software



Left & Right IR sensors

Modify Avoider so that it can detect an object "further" away on the track using the Center IR sensor and then STOP.

Also can you make the bot swing around completely and then follow the track in the reverse direction?



Possible Hardware Solution