|  |  |
| --- | --- |
| MIXED REALITY ROBOTICS |  Commands and Parsing Feedback |

Once in the Terminal, you have access to most of Pleo’s sensors. The command “***sensor show***” will open up a list of all the sensors and preset values. Further exploration of the “***help***” command will explain who to alter and change these values.

This is what the sensor **show screen** will look like:

**For example, this row here shows the Battery sensor information. The Trigger level is 25, because the robot will shut down if the battery value is below 25%. This battery, however, is currently at a value of 75%.**



The most important columns in this chart are:
“Sensor” – which states the sensor number
“Enabled” – which states whether the sensor is on “1” or off “0”
“Value” – which states the current set value of the sensor, especially important for battery
“Name” – which states the name of the sensor

The rest of the information, such as “Trig Level” – which is the set value at which the sensor is triggered – can be taken into account, but aren’t necessary important to programming the Pleo.

However, just gaining access to the sensors does not provide real time feedback as to what sensors the Pleo is currently feeling. In order to do this, a PAWN program must be written, compiled onto an SD card and then inserted into the Pleo. This PAWN program commands the Pleo to return sensor feedback to the Terminal screen.

The Pleo Development Tool Kit that is needed to install the appropriate driver for Pleo to connect to a computer also comes with many “.p” PAWN programs that can be used to control the Pleo and access information. These files are a key part to getting sensor data from the Pleo. In order to print sensor data onto the screen, the only thing that must be altered is “sensors.p” file which can be found in the Tool Kit in the folder called bin. Replace that program code with this code and then compile it onto an SD card:

*// Very simple sensors.p example. Add code to on\_sensor for those*

*// sensors you would like to respond to.*

*// save space by packing all strings*

*#pragma pack 1*

*#include "Log.inc"*

*#include "Script.inc"*

*#include "Sensor.inc"*

*#include "Sound.inc"*

*#include "joints.inc"*

*#include "Joint.inc"*

*#include "Motion.inc"*

*#include "motions.inc"*

*#include "sounds.inc"*

*public init()*

*{*

*print("sensors:init() enter\n");*

*print("sensors:init() exit\n");*

*}*

*public on\_sensor(time, sensor\_name: sensor, value)*

*{*

*new name[32];*

*sensor\_get\_name(sensor, name);*

*printf("sensors:on\_sensor(%d, %s, %d)\n", time, name, value);*

*}*

*public close()*

*{*

*print("sensors:close() enter\n");*

*print("sensors:close() exit\n");*

*}*

The sensor feedback looks like a constant stream of sensor information. In order to pause the stream press “control” + C:



**For example, this row here shows the SENSOR\_CARD\_DETECT sensor feedback. This shows whether an SD Card is detected by the Pleo – “1” if yes and “0” if no. It should return “1” because an SD card is needed to access this feedback screen.**

**Sensor Evaluation**

Most of Pleo’s sensors fall into one of five categories: touch, motor, sound, camera and infrared. The most reliable and useful of these are the touch and motor sensors.

Infrared Sensors play a large role in leaf and other Pleo detection, but they are very difficult to access and understand. Moreover, the infrared readings are rarely reliable.

Pleo’s camera can take photos and has some blob detection. However, this camera is very limited because it cannot receive real time images (all images are saved to the SD card) and the blob detection is incredibly unreliable.

The robot’s speakers allow the behaviors to be complemented by emotion evoking sounds, which can be very useful. But, as far as sound detection goes, the Pleo does not recognize sounds very well because of high sound wave interference from the environment.

Motors and Touch sensors, however, work well together in creating emotion evoking actions and behaviors that can be implemented into robot education of elementary school children.

A sensor that will not aid much in programming, but is important to keep track of is the Battery Sensor complex, which is described in the Miscellaneous Sensor category.