Smarty Development Plan

# Introduction

## Goal

Develop an attractive, fun, accessible, robust, and cheap robotics education kit based on well-known Scratch 2 graphical programming environment.

## Versions

Version 1 :

* smarty brick connected to the development PC via USB or Bluetooth
* the code runs on the PC
* the devices are directly connected to the smarty brick
* the ICD is ASCII

Version 2 :

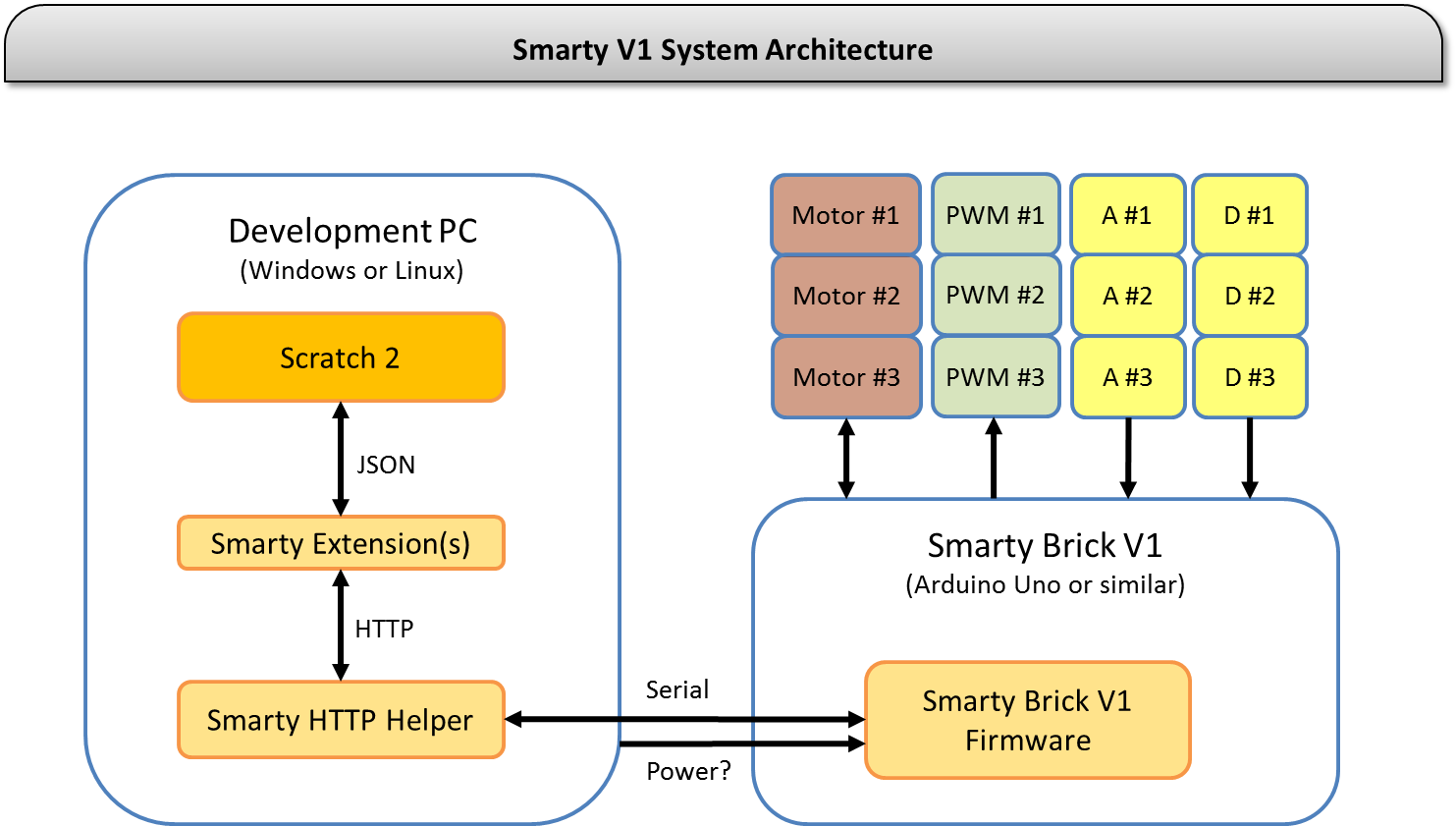
* same as V1 except the devices are smarties, and connect to the smarty brick via RF
* the smarty brick turns essentially into a gateway for the wireless smarties
* the code still runs on the PC
* the ICD is binary

Version 3 :

* smarty brick becomes smarter: based on Raspberry Pi instead of Arduino
* the dev env (Scratch 2) can run on the PC or on the smarty brick
* the code runs on the smarty brick only, which can be disconnected from the PC, thus creating a fully wireless autonomous robotics platform.

# Version 1

## System Architecture



## Key Characteristics

-

-

-

## Hardware Description

* PC with Windows or Linux (Mac?) for development and runtime
* Arduino Uno or compatible as smarty (dumby?) brick
* Adafruit motor shield v1.2, providing control for:
  + 2 servos
  + 4 DC motors or 2 DC motors+1 stepper or 2 steppers
  + 6 analog pins (can be configured as analog inputs or digital i/o)
  + 1 digital i/o
* devices directly connected to the motor shield

## Interface Description

### Extension Interface

Type : HTTP

…

### Smarty Brick Interface

#### Definition

Type: Serial

Baudrate: 115200

Parity: None

Data Bits: 8

Stop Bits: 1

#### Protocol Description

The protocol consists of ASCII strings, both for commands and for periodic status.

All strings are newline-terminated (with ‘\n’, ASCII code 10).

All strings start with the synchronization word: “#smarty#”.

Commands start with a letter which specifies the device type, followed by a two-digit identifier of the device, where the identifiers for each type start at 1.

Arguments in a message are :-separated.

#### Commands

**Motors**: #smarty#mNN:[+SSS/-SSS/0]<:TTTT>\n

where: NN is the motor’s 2-digit identifier (01, 02, etc.)

+SSS sets the motor speed in one direction, where SSS is 001 to 255 (translates

to PWM command, where 001 is minimum speed and 255 maximum speed)

-SSS sets the motor speed in the other direction

Instead of +SSS or –SSS the argument can be a single 0, in this case the motor is

stopped and the smarty does not look for any other characters in this command.

<:TTTT> is an optional 4-digit argument which sets an internal timer in the

smarty to stop the motor after TTTT/10 secs. (e.g. if TTTT is 1234 then after

123.4 secs)

**PWM**: #smarty#pNN:SSS<:TTTT>\n

where: NN is the output’s 2-digit identifier (01, 02, etc.)

SSS sets the PWM output value, from 000 to 255

<:TTTT> is an optional 4-digit argument identical to the motor command

#### Status

The smarty brick sends a periodic status at 100Hz, independent on the reception of commands.

The report summarizes the statuses from all devices connected to the smarty brick, written back-to-back and comma-separated.

Each device status has a uniform format: <device type><device id>:<device status>.

where: <device type> can be ‘m’ for motor, ‘d’ for digital input, ‘a’ for analog input

<device id> is the 2-digit device identifier like in the commands

<device status> is a string describing the device status, usually a number

Motor status: mNN:[+SSS/-SSS]C

[+SSS/-SSS] is the commanded motor speed & direction. The reported

value can be different from the command due to timer <,TTTT>.

C is a rough estimate of current consumption, on a scale from 0 (no

current) to 9 (over-current). If not applicable, C is ‘x’.

Digital input: dNN:[0/1]

Analog input: aNN:AAAA

AAAA is the raw read value by the A2D (0 to 1024)

Example status:

#smarty#m01:+1505,m02:-2559,

d01:0,d02:1,a01:0004,a02:1000,a03:0567\n

## Software Description

(Aviv/Eyal/Gleb)

## Firmware Description

The smarty brick is based on Arduino UNO.

Command parsing:

* the loop() function waits for \n-terminated strings, of maximum length 50.
* then each string is parsed, starting by the sync word (“#smarty#”) then the one-character opcode (‘m’ or ‘p’ – can be extended in the future)

Status report:

## System Integration Plan

# Version 2

## System Architecture

## Key Characteristics

## Hardware Description

## Interface Description

### Extension Interface

### Smarty Brick Interface

#### Definition

Type: Serial

Baudrate: 115200

Parity: None

Data Bits: 8

Stop Bits: 1

#### Protocol Description

The protocol consists of binary messages, both for commands and for periodic status.

The comBrick acts as a gateway and does not change the content of the messages, in both directions, between the PC Helper and the remote smarties.

Each message contains a source id (1 byte), destination id (1 byte), opcode (1 byte) and content (7 bytes) for a total of 10 bytes fixed length messages.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 | byte 8 | byte 9 |
| src id | dest id | opcode | content | | | | | | |

in general:

- NULL byte means 0x00

- all commands opcode start with C, motor control starts with E, sensors data update starts with D

-<uint> byte means a value between 0 and 255.

-<bool>byte means a zero value or non-zero value. zero value (0x00) means false or low, non-zero value (we will use 0x01) means true or high.

#### Commands

##### Generic Command

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 | byte 8 | byte 9 |
| src id | dest id | 0xC0 | cmd# | param1 | param2 | param3 | param4 | param5 | param6 |

params are optional. NULL if unused

##### SetAnalog Command

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 | byte 8 | byte 9 |
| src id | dest id | 0xCA | analog# | analog val | analog# | analog val | analog# | analog val | NULL |

only first (analog#, analog value) pair is mandatory. rest is optional. if unused put 0xFF in the analog# and NULL in the value.

##### SetDigital Command

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 | byte 8 | byte 9 |
| src id | dest id | 0xCD | digital# | digital val | digital# | digital val | digital# | digital val | NULL |

only first (digital#, digital value) pair is mandatory. rest is optional. if unused put 0xFF in the digital# and NULL in the value

##### DC Motor Command

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 | byte 8 | byte 9 |
| src id | dest id | 0xE0 | speed update | speed value | power update? | power value? | direction update | direction value | NULL |

#### Status

## Software Description

(Aviv/Eyal/Gleb)

## Firmware Description

## System Integration Plan

# Version 3