```
' {$STAMP BS2}
' {$PBASIC 2.5}

' Gimme Sugar v1.1 - A Basic Stamp software for gesture controlled sugar dispencer
' Copyright (C) 2007 Anna Keune, Jari Suominen
' For more details: mlab.taik.fi/paja

' This program is free software: you can redistribute it AND/OR modify
' it under the terms of the GNU General Public License as published by
' the Free Software Foundation, either version 3 of the License, OR
' (at your option) any later version.

' This program is distributed in the hope that it will be useful,
' but WITHOUT ANY WARRANTY; without even the implied warranty of
' MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.  See the
' GNU General Public License FOR more details.

' You should have received a copy of the GNU General Public License
' along with this program.  IF NOT, see <http://www.gnu.org/licenses/>.

' Version Log:
' v0.1
' This version just opens the valve completely and shuts it after some amount of time.
' v0.2
' This version can pour in two different ways: full open with sharp angle and half open
'  with low angle.
' v0.3
' This version allready had a great led light show
' v0.4
' Optimized version
' v0.5
' Bugfixes and more optimization
' v0.6
' Cleaning up the Code
' v0.7
' Final tweaking of the parameters
' v1.0
' The perfect Lego prototype -version
' v1.1
' This is the Demo Day -version. Tweaked for new dispencer/shaker hardware. Added
'  support for vibra-motor and second light sensor. Valve positions updated for the new
'  valve.

' ---[ I/O Definitions ]---
pLed1         PIN  3
pLed2         PIN  4
pLed3         PIN  5
pYin          PIN  9   ' accelerometer (MX2125)
pLightSensor  PIN  15
pLightSensorr PIN  14
pMotor        PIN  12  ' servo motor (MX-400)
pVibra        PIN  8   ' vibration motor (from Nokia 3310)
```

```
' ---[ Variables ]---
wYPulse         VAR  Word    ' raw data from accelerometer
wYGravity       VAR  Word    ' Y axis gravity
wResult         VAR  Word

sugarInThePipe  VAR  Word
pouring         VAR  Bit

ledCounter      VAR  Word
ledState        VAR  Word
steps           VAR  Word

' ---[ Constants ]---
cPulseState    CON  1
cScale         CON  $200                ' 2.0 us per unit

CENTER         CON  30000    'IS THIS ACTUALLY NEEDED? Yes it is.

'Constants for the valve servo
FULLOPEN        CON  890 '700
HALFOPEN        CON  800 '580
CLOSED          CON  600 '440

MAXSUGARINTHEPIPE    CON  50
FULLOPENSUGARCONSUMEDINITERATION  CON  5
HALFOPENSUGARCONSUMEDINITERATION  CON  2

ANGLEFORFULLOPEN    CON  65100
REFILLANGLE         CON  800

LEDSTATECHANGE      CON  50

' ---[ Initialization ]---
DEBUG CLS, "Gimme Sugar. Copyright (C) 2007 Anna Keune, Jari Suominen", CR
DEBUG "This program comes with ABSOLUTELY NO WARRANTY", CR
DEBUG "This is free software, AND you are welcome TO redistribute it", CR
DEBUG "under certain conditions; read source code for details.", CR, CR

ledState = 0
pouring = 0 'false
sugarInThePipe = 0
ledCounter = LEDSTATECHANGE + 1

' ---[ Main Code ]---
Main:
DO
  GOSUB Read_Accelerometer
  wYGravity = wYGravity * -1 ' sensor was installed inside the shaker upside down,
                             ' this turns it back again.
  'DEBUG "y: ", DEC wYGravity, CR
  GOSUB Control_Valve
```

```
   GOSUB Update_Pouring_State
   GOSUB Do_The_Lightshow
LOOP
END

' ---[ Subroutines ]---
Control_Valve:
 IF sugarInThePipe = 0 THEN
       PULSOUT pMotor, CLOSED
       HIGH pVibra
 ELSEIF sugarInThePipe > 0 THEN
   IF pouring THEN
     IF wYGravity < ANGLEFORFULLOPEN THEN
       PULSOUT pMotor, FULLOPEN
       sugarInThePipe = sugarInThePipe - FULLOPENSUGARCONSUMEDINITERATION
     ELSE
       PULSOUT pMotor, HALFOPEN
       sugarInThePipe = sugarInThePipe - HALFOPENSUGARCONSUMEDINITERATION
     ENDIF
     IF sugarInThePipe > MAXSUGARINTHEPIPE THEN
       sugarInThePipe = 0
     ENDIF
       DEBUG "Sugar in the pipe: ", DEC sugarInThePipe, CR
   ELSE
     PULSOUT pMotor, CLOSED
   ENDIF
 ENDIF
RETURN

Update_Pouring_State:
 IF wYGravity > CENTER THEN 'NOTE: negative numbers without signs are big ones (>
CENTER)
       IF pouring = 0 THEN
         IF sugarInThePipe > 0 THEN
            DEBUG "Gimme Sugar!", CR
            pouring = 1
       LOW pVibra
         ELSE
            DEBUG "You have to refill!", CR
         ENDIF
       ENDIF
   ELSE ' wyGravity < CENTER
       IF pouring THEN
     HIGH pVibra
         pouring = 0
         DEBUG "Pouring ended!", CR
       ENDIF
       IF wYGravity > REFILLANGLE THEN
         IF sugarInThePipe < MAXSUGARINTHEPIPE THEN
            sugarInThePipe = MAXSUGARINTHEPIPE
            DEBUG "Refilled!", CR
            LOW pVibra
```

```
            PAUSE 250          'Small buzz indicating movement of sugar when
            HIGH pVibra        'the shaker is set straight
          ENDIF
        ENDIF
    ENDIF
RETURN

Do_The_Lightshow:
 wResult = pLightSensor ' 0 or 1
 IF wResult = 0 THEN
   wResult = pLightSensorr ' Check the other sensor incase the user has taken
                           ' the shaker in hand in some odd position
 ENDIF
 'DEBUG "light", DEC wResult, CR
 IF wResult = 1 THEN
       IF wYGravity < REFILLANGLE THEN
         ledCounter = ledCounter + 6*(REFILLANGLE-wYGravity)/LEDSTATECHANGE
         'DEBUG DEC 6*(REFILLANGLE-wYGravity)/LEDSTATECHANGE, CR
       ELSEIF wYGravity > CENTER THEN
          ledCounter = ledCounter + LEDSTATECHANGE + 1 'forcing ledstate change on
every iteration
       ENDIF
   IF sugarInThePipe = 0 THEN
         ledCounter = 0 'forcing ledstate to remain the same in every iteration
   ENDIF
       IF ledCounter > LEDSTATECHANGE THEN
         ledState = ledState + 1
         IF ledState > 2 THEN
            ledState = 0
         ENDIF
          ledCounter = 0
       ENDIF
   IF ledState = 0 THEN    ' twinkle twinkle little star...
           HIGH pLed1
           LOW  pLed2
           LOW  pLed3
       ELSEIF ledState = 1 THEN
           HIGH pLed2
           LOW  pLed1
           LOW  pLed3
       ELSEIF ledState = 2 THEN
           HIGH pLed3
           LOW  pLed1
          LOW  pLed2
        ENDIF
   ELSE      'Shaker not in hand
       LOW  pled1
       LOW  pled2
       LOW  pled3
 ENDIF
RETURN
```

```
Read_Accelerometer:
 PULSIN pYin, cPulseState, wYPulse
 wYPulse = wYPulse */ cScale
 wYGravity = ((wYPulse / 10) - 500) * 8
RETURN
```