//Collects current power consumption from smart meter via ZigBee or from current clamp around Service point

//Reports current consumption back to Agent and onto server

//Retrieves current community consumption averages from Agent(cloud IOT service) and displays a three level status back to the user

//Levels are based on real time community averages. With an LED face lighting up according to their comparative usage

meterReading <-0

pot <- hardware.pin5

redArray <- hardware.pin1

greenArray <- hardware.pin2

smile <- hardware.pin8

sad <- hardware.pin9

//#defined values are not available in Squirrel code

maxSupply <- 17000 //17KWh

//periodic push of reading to agent

function UpdateCommunity(){

local meterReading = ReadMeter()

agent.send("usage", meterReading)

imp.wakeup( 6 , UpdateCommunity)

}

function UpdateDisplay(){

local communityAverage = 8500

local lowerLimit = communityAverage \* 0.8 //Set 80% less than community average

local upperLimit = communityAverage \* 1.2 //Set 20% over community average

local localLevel = ReadMeter();

if(localLevel <= lowerLimit) //Positive

{

//server.log("Positive")

redArray.write(0)

greenArray.write(0.3)

smile.write(0.3)

sad.write(0)

}

else if(localLevel >= upperLimit) //Negative

{

//server.log("Negative")

redArray.write(1)

greenArray.write(0)

smile.write(0)

sad.write(1)

}

else //Neutral

{

//server.log("Neutral")

redArray.write(1)

greenArray.write(0.1)

smile.write(0)

sad.write(0)

}

imp.wakeup( 0.3 ,UpdateDisplay) //if agent is unresponsive this routine will continue to update the display based on old data

}

function ReadMeter(){

//This should be a power sensing system on the service connection

//Alternately a serial link with a ZigBee module to interface with

//A smart meter to retrieve consumption in kWh provided it is granluar

local adcReading = 0

local load = 0.00

adcReading = pot.read() //Sample ADC input

//server.log("ADC")

//server.log(adcReading)

load = (adcReading \* maxSupply) / 65536 //Set load as a percentage of maximum load

//server.log("LOAD ")

//server.log(load)

return load

}

function configureIMP(){

//set IO as analogue input for mock consumption

pot.configure(ANALOG\_IN)

//Set IO as Outputs for status display

redArray.configure(DIGITAL\_OUT)

greenArray.configure(PWM\_OUT, 1.0 / 400.0, 0.0)

smile.configure(PWM\_OUT, 1.0 / 400.0, 0.0)

sad.configure(DIGITAL\_OUT)

}

server.log("Imp Starting Up")

configureIMP()

UpdateDisplay()

UpdateCommunity()

//agent.on("update", UpdateDisplay)