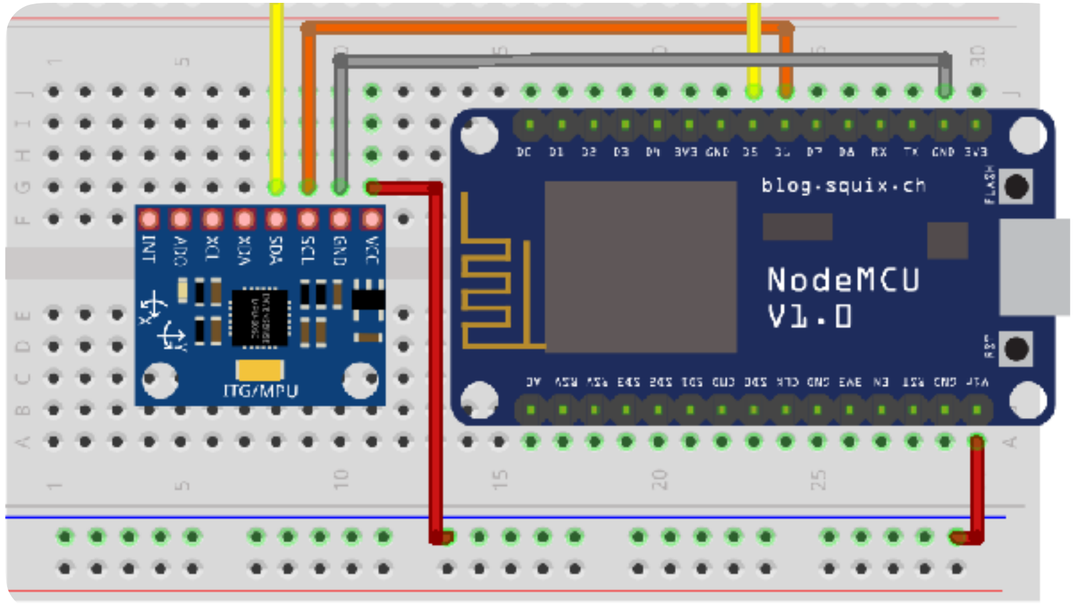
Trabalho NodeMCU

SISTEMAS REATIVOS

Materiais

- Node MCU

- Acelerômetro MPU 6050



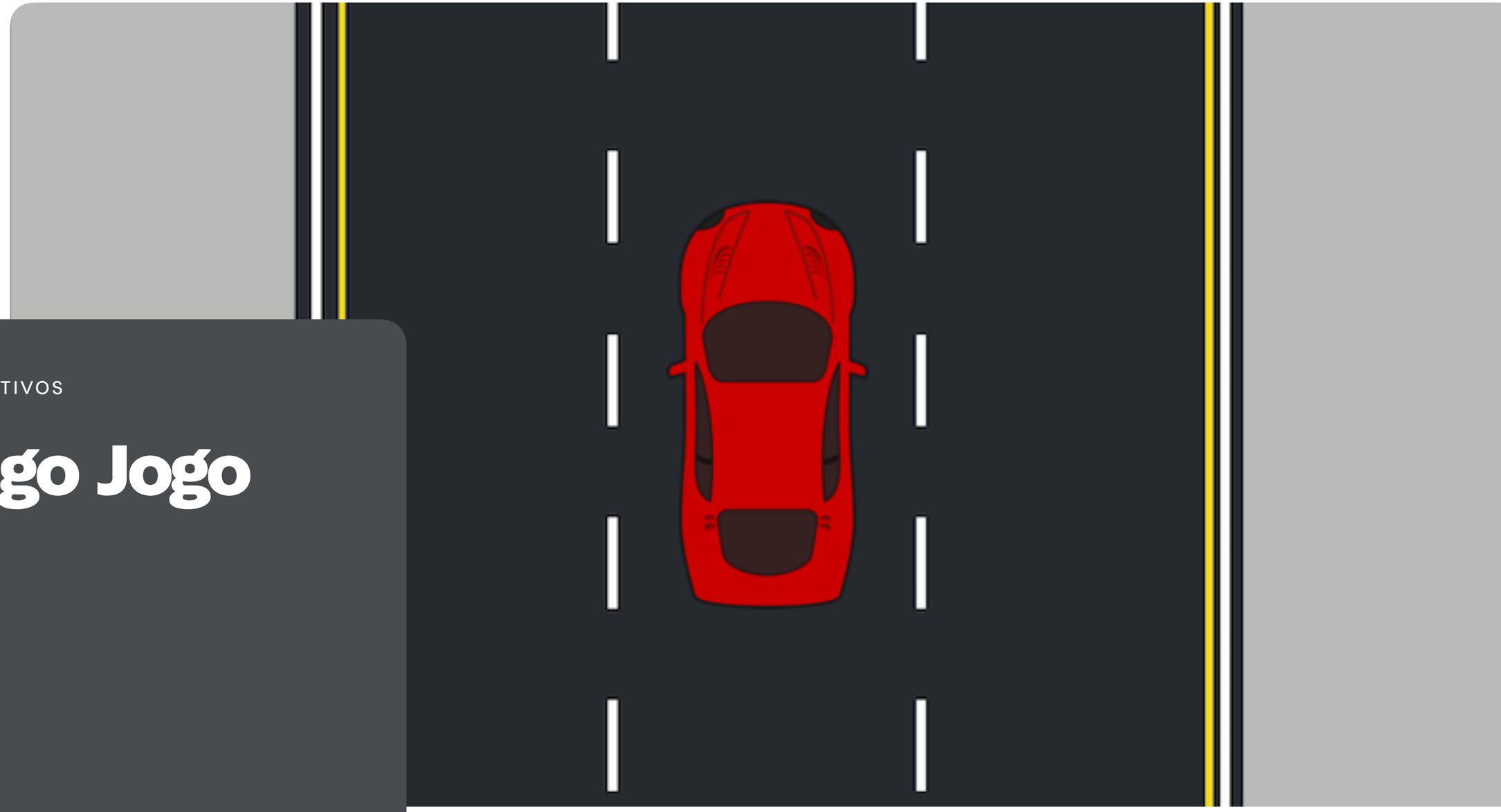
Overview

- Simulação de um carro em Lua LOVE
- Acelerômetro MPU 6050 como controle
  - Rotação do carro - inclinação para esquerda / direita
  - Aceleração do carro - inclinação para frente (acelera) / trás (desacelera)
- Comunicação via wifi (servidor mqtt)

DEMO

SISTEMAS REATIVOS

Código Jogo



```lua
function love.draw()
    -- Desenha uma imagem de uma estrada 3 vezes (posição das imagens é atualizada
    -- de acordo com a velocidade no eixo y para criar uma ilusão de movimento)
    love.graphics.draw(road, 0, dy)
    love.graphics.draw(road, 0, -y + dy)
    love.graphics.draw(road, 0, y + dy)

    -- Hitbox & desenha o carro
    if car.x < 107 then
        car.x = 107
    elseif car.x > (x - 107) then
        car.x = (x - 107)
    end
    if car.y < 100 then
        car.y = 100
    elseif car.y >  (y - 100) then
        car.y = (y - 100)
    end
    love.graphics.draw(carImage, car.x, car.y, math.rad(car.rotation), 0.5)
end
```

SISTEMAS REATIVOS

Inicialização

```lua
function love.load()
    ACCELERATION = 2
    dy = 0

    -- Carrega as imagens
    road = love.graphics.newImage("road.png")
    carImage = love.graphics.newImage("car.png")

    -- Inicializa o carro no centro da tela
    initializeCar()

    -- Inicializa a aceleração e a rotação do carro
    joystick = {
        acceleration = 0,
        rotation = 0,
    }
    initializeJoystick()
end
```

```lua
local x, y = love.graphics.getDimensions()

function initializeCar()
    car = {
      x = x/2 - carImage:getWidth()/2,
      y = y/2 - carImage:getHeight()/2,
      xvel = 0,
      yvel = 0,
      rotation = 0
    }
end

function initializeJoystick()
    mqtt = require("mqtt_library")

    -- Atualiza a rotação / aceleração do joystick com o valor enviado
    updateDirection = function(topic, msg)
        if topic == "up" then
            joystick.acceleration = msg
        elseif topic == "down" then
            joystick.acceleration = -msg
        elseif topic == "left" then
            joystick.rotation = -msg
        else
            joystick.rotation = msg
        end
    end

    local __joystick = mqtt.client.create("127.0.0.1", 1883, updateDirection)

    __joystick:connect("love")
    __joystick:subscribe({"up", "down", "left", "right"})

    joystick.joystick = __joystick
end
```

SISTEMAS REATIVOS

Preparação

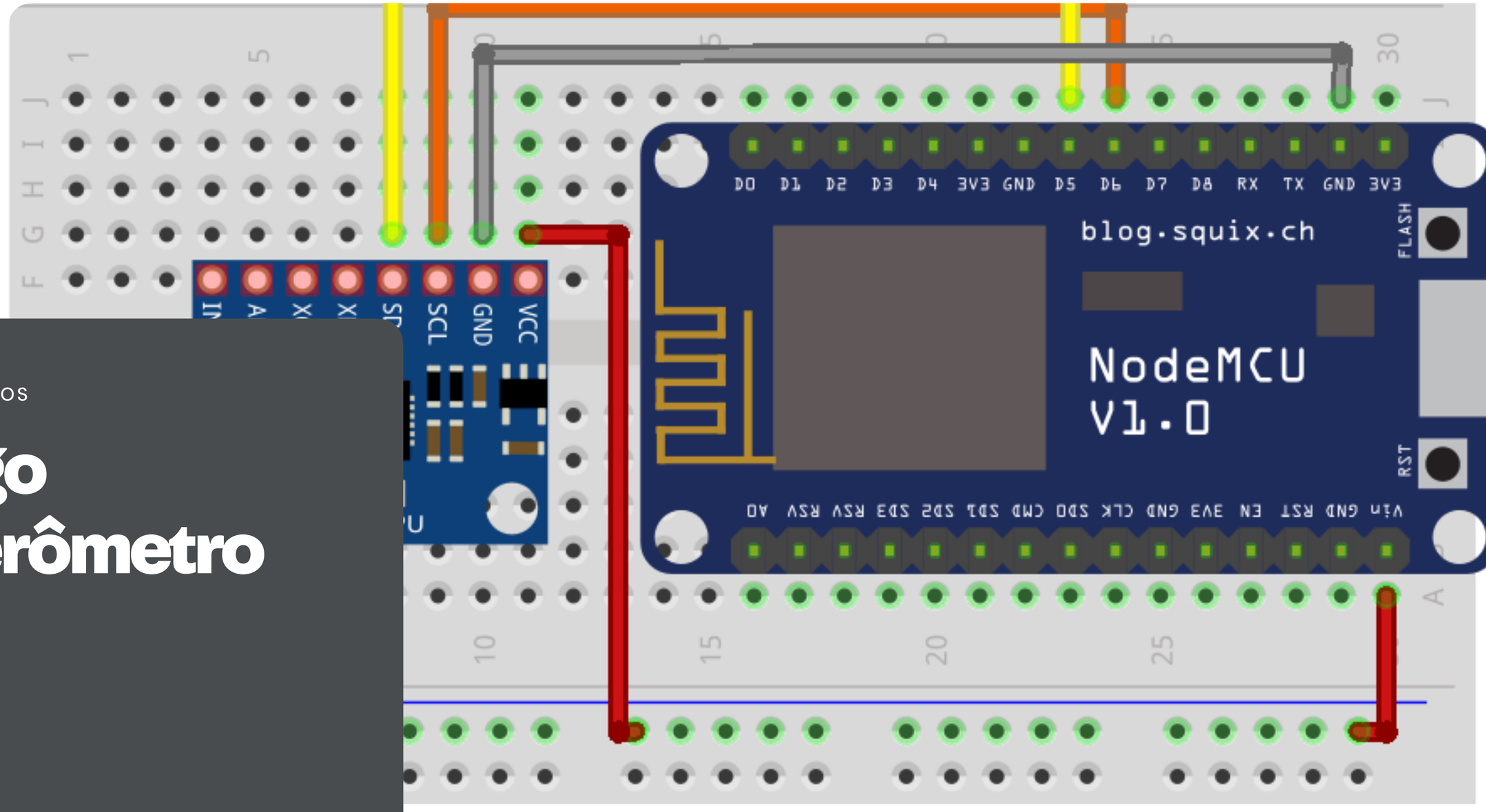
```lua
function love.update(dt)
    if joystick.acceleration ~= 0 then
     local increment = joystick.acceleration
     ACCELERATION = ACCELERATION + increment > 10 and 10 or ACCELERATION + increment
    end

    -- Determina a velocidade no eixo x
    car.xvel = car.xvel + ACCELERATION*dt * math.cos(car.rotation)
    if (car.rotation < 0 and car.xvel > 0) or (car.rotation > 0 and car.xvel < 0) then
        car.xvel = -car.xvel
    end

    -- Determina a posição no eixo x do carro
    car.x = car.x + car.xvel*dt


    -- Obtem a rotação do carro
    car.rotation = joystick.rotation

    -- Determina a velocidade do carro no eixo y baseado na sua rotação
    if (car.rotation == 0) then
        car.yvel = car.yvel + ACCELERATION*dt
    else
        car.yvel = car.yvel + ACCELERATION*dt * math.abs(math.sin(car.rotation))
    end


    car.xvel = car.xvel * 0.99
    car.yvel = car.yvel * 0.99


    -- Atualiza a posição do background de acordo com a velocidade no eixo y
    dy = math.abs(dy + car.yvel) > road:getHeight() and 0 or dy + car.yvel

end
```

SISTEMAS REATIVOS

Atualização

SISTEMAS REATIVOS

Código Acelerômetro



```lua
dev_addr = 0x68 --104
bus = 0
sda, scl = 3, 4

function init_I2C()
  i2c.setup(bus, sda, scl, i2c.SLOW)
end

function init_MPU(reg,val)  --(107) 0x6B / 0
    write_reg_MPU(reg,val)
end

function write_reg_MPU(reg,val)
  i2c.start(bus)
  i2c.address(bus, dev_addr, i2c.TRANSMITTER)
  i2c.write(bus, reg)
  i2c.write(bus, val)
  i2c.stop(bus)
end

function read_reg_MPU(reg)
  i2c.start(bus)
  i2c.address(bus, dev_addr, i2c.TRANSMITTER)
  i2c.write(bus, reg)
  i2c.stop(bus)
  i2c.start(bus)
  i2c.address(bus, dev_addr, i2c.RECEIVER)
  c=i2c.read(bus, 1)
  i2c.stop(bus)
  return c
end
```

SISTEMAS REATIVOS

Inicialização

```lua
function status_MPU(dev_addr)
      i2c.start(bus)
      c=i2c.address(bus, dev_addr ,i2c.TRANSMITTER)
      i2c.stop(bus)
      if c==true then
         print(" Device found at address : "..string.format("0x%X",dev_addr))
      else print("Device not found !!")
      end
end

function check_MPU(dev_addr)
   print("")
   status_MPU(0x68)
   read_reg_MPU(117) --Register 117 - 0x75
   if string.byte(c, 1) == 104 then print(" MPU6050 Device answered OK!")
   else print("  Check Device - MPU6050 NOT available!",c,string.byte(c, 1))
        return
   end
   read_reg_MPU(107) --Register 107
   if string.byte(c, 1)==64 then print(" MPU6050 in SLEEP Mode !")
   else print(" MPU6050 in ACTIVE Mode !")
   end
end

loadfile("mqtt.lua")
init_I2C()
check_MPU(0x68)
init_MPU(0x6B,0)

read_MPU_raw()

tmr.alarm(0, 1000, 1, function() read_MPU_raw() end)
```

SISTEMAS REATIVOS

Inicialização

```lua
local function roundToFirstDecimal(num)
  return math.floor(num * 10) / 10
end

function get_direction()
  if (Ay >= 3) then
    -- Turn left. Intensity: 4-Ay
    publish("left", 4-Ay)
  elseif (Ay > 0.1) then
    -- Turn right. Intensity: Ay
    publish("right", Ay)
  end
end

function get_acceleration()
  if (Ax >= 3) then
    -- Speed up. Intensity: 4-Ax
    publish("up", 4-Ax)
  elseif (Ax > 0.2) then
    -- Slow down. Intensity: Ax
    publish("down", Ax)
  end
end
```

SISTEMAS REATIVOS

Funções auxiliares

```lua
function read_MPU_raw()
  i2c.start(bus)
  i2c.address(bus, dev_addr, i2c.TRANSMITTER)
  i2c.write(bus, 59)
  i2c.stop(bus)
  i2c.start(bus)
  i2c.address(bus, dev_addr, i2c.RECEIVER)
  c=i2c.read(bus, 14)
  i2c.stop(bus)

  Ax=bit.lshift(string.byte(c, 1), 8) + string.byte(c, 2)
  Ay=bit.lshift(string.byte(c, 3), 8) + string.byte(c, 4)
  Az=bit.lshift(string.byte(c, 5), 8) + string.byte(c, 6)

  Ax=roundToFirstDecimal(Ax/16384)
  Ay=roundToFirstDecimal(Ay/16384)
  Az=roundToFirstDecimal(Az/16384)

  get_direction()
  get_acceleration()

  return c, Ax, Ay, Az
end
```

SISTEMAS REATIVOS

Leitura e conversão dos dados