Arduino+Senzor Vibratii Brick + MODIO2 + Telefon Vechi => Sistem de control la distanta peste GSM

In acest tutorial iti propun ideea de a folosi un telefon vechi (setat pe vibratii) impreuna cu un senzor de vibratii brick. Atunci cand apelezi telefonul, acesta vibreaza, senzorul de vibratii sesizeaza acest lucru, si ca raspuns Arduino actioneaza un releu. Care releu poate actiona orice vrei tu. De exemplu, o pompa de irigantii. Sau un reflector. Sau orice altceva ai nevoie.

Poti folosi orice telefon din generatia mai veche, cu conditia sa dispuna de apel prin vibratii. Senzorul il vei conecta la o placa Arduino UNO. Pentru exemplul de mai jos, ca sarcina pentru releu vom folosi un led brick. Evident ca tu il poti inlocui cu orice sarcina de putere mare.

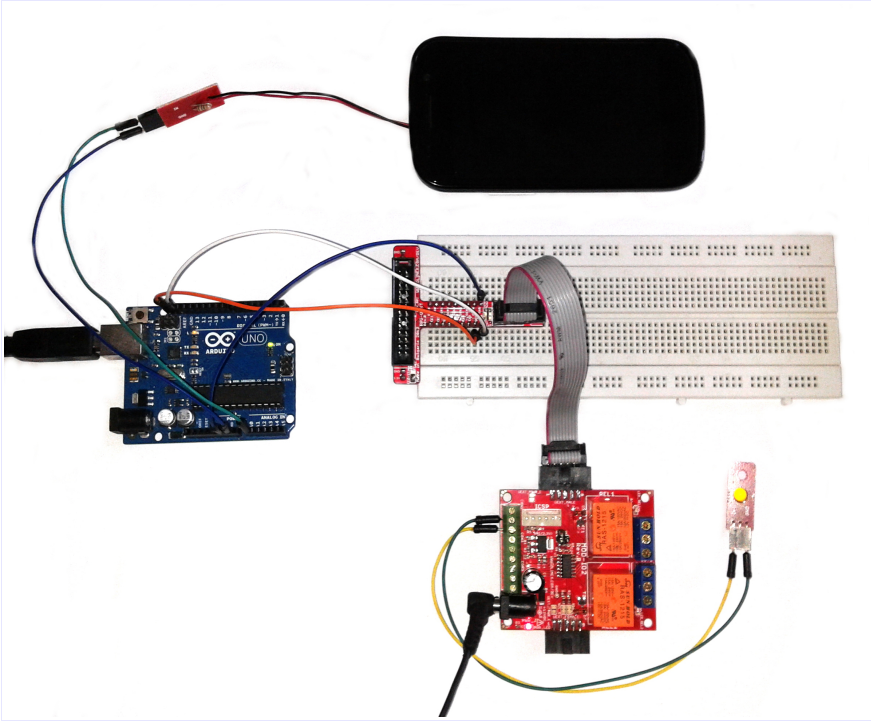
Componentele necesare:

- Arduino UNO.
- MOD-IO2.
- Senzor de vibratii brick.
- Adaptor Rpi-UEXT.
- Breadboard.
- Sursa de alimentare pentru MOD-IO2 (12V).
- LED brick.
- Fire pentru conexiuni.
- Telefon.

Tabelul de conexiuni:

Arduino UNO SCL	Rpi-UEXT SCL
Arduino UNO SDA	Rpi-UEXT SDA
Arduino UNO A0	Senzor vibratii brick IN
Arduino UNO GND	Rpi-UEXT GND
Arduino UNO GND	Senzor vibratii brick GND
Led brick IN	MOD-IO2 GPIO 0
Led brick GND	MOD-IO2 GND

Realizeaza conexiunile conform tabelului. Vei obtine o schema asemanatoare cu cea din imaginea de mai jos:



Incarca pe Arduino cod de mai jos. Pentru testarea sistemului, nu trebuie decat sa apelezi telefonul si sa observi cum MOD-IO2 comuta releul (si led-ul se aprinde).

Totusi inainte de a incarca sketch-ul, este necesar sa realizezi o modificare in mediul Arduino. Magistrala I2C a placii Arduino lucreaza la frecventa de 100Khz dar MOD-IO2 lucreaza cu succes doar la 50Khz. Vei modifica frecventa placii Arduino din fisierul de mai jos.

arduino-1.0.2/libraries/Wire/utility/twi.h

Deschide fisierul **twi.h** (cu Notepad, de exemplu) si cauta liniile:

```
#ifndef TWI_FREQ
#define TWI_FREQ 100000L
#endif
```

Schimba valoarea 100000L in 50000L.

Cand vrei sa abordezi alte proiecte, nu uita sa schimbi la loc valoarea.

```
#include <Wire.h>
int led = 0;
int whichRelay = 0;

const int contor1=5; // secunde
const int contor2=5; // secunde
const int nivelVibratii = 1; // nivelul de vibratii

void setup() {
  Serial.begin(9600);
  Wire.begin();
  setareGPIO();
  setGPIOState(0,0);
  setRelayState(0,0);
}


void loop() {
  int val = maxim(0, 100);
  if (val > nivelVibratii) {
   setGPIOState(0,1);
   setRelayState(0,1);
   delay(contor1*1000);
   setGPIOState(0,0);
   setRelayState(0,0);
   delay(contor2*1000);
  }
}

int maxim(int pin, int count) {
 int valoareMaxima = 0;
 for (int i = 0; i < count; i++) {
   valoareMaxima = max(analogRead(pin), valoareMaxima);
  }
  return valoareMaxima;
}

void setareGPIO() {
 Wire.beginTransmission(0x48);
 Wire.write(0x02);
 Wire.write(0xA0);
 Wire.write(0x01);
 Wire.write(0x00);
 Wire.endTransmission();
}

void setGPIOState(int gpio, int state) {
```

```
 if (state == 1) bitSet(led, gpio);
 if (state == 0) bitClear(led, gpio);
 Wire.beginTransmission(0x48);
 Wire.write(0x02);
 Wire.write(0xA0);
 Wire.write(0x02);
 Wire.write(led);
 Wire.endTransmission();
}

void setRelayState(int relay, int state) {
 if (state == 1) bitSet(whichRelay, relay);
 if (state == 0) bitClear(whichRelay, relay);
 Wire.beginTransmission(0x48);
 Wire.write(0x02);
 Wire.write(0xA0);
 Wire.write(0x40);
 Wire.write(whichRelay);
 Wire.endTransmission();
}
```

In program sunt definite 2 constante: contor1 si contor2, constante responsabile cu temporizarea releului. Daca cele 2 constante au valori mai mari atunci temporizarea releului va fi la fel de mare.

A treia constanta stabileste nivelul de vibratii la care va reactiona Arduino. Daca nivelul este depasit in bucla loop() atunci Arduino comanda placa MOD-IO2.

O idee simpla (si ieftina !) de a comanda un anumit consumator de la distanta, prin telefonul mobil.