

```
#include <Keypad.h>

int datasens;

#define pinsens 11

const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
    {'1','2','3'},
    {'4','5','6'},
    {'7','8','9'},
    {'*','0','#'}
};

byte rowPins[ROWS] = {2, 3, 4, 5}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {6, 7, 8}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
int digit;
int key1;
int key2;
int key3;
int key4;
int kode;
int pres;
```

```

boolean blink = false;
boolean ledPin_state;
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(A5, A4, A3, A2, A1, A0);

#include "Arduino.h"
#if !defined(SERIAL_PORT_MONITOR)
    #error "Arduino version not supported. Please update your IDE to the latest
version."
#endif

#if defined(__SAMD21G18A__)
    // Shield Jumper on HW (for Zero, use Programming Port)
    #define port SERIAL_PORT_HARDWARE
    #define pcSerial SERIAL_PORT_MONITOR
#elif defined(SERIAL_PORT_USBVIRTUAL)
    // Shield Jumper on HW (for Leonardo and Due, use Native Port)
    #define port SERIAL_PORT_HARDWARE
    #define pcSerial SERIAL_PORT_USBVIRTUAL
#else
    // Shield Jumper on SW (using pins 12/13 or 8/9 as RX/TX)
    #include "SoftwareSerial.h"
    SoftwareSerial port(12, 13);
    #define pcSerial SERIAL_PORT_MONITOR
#endif

```

```
#include "EasyVR.h"
```

```
EasyVR easyvr(port);
```

```
//Groups and Commands
```

```
enum Groups
```

```
{
```

```
    GROUP_0 = 0,
```

```
    GROUP_1 = 1,
```

```
};
```

```
enum Group0
```

```
{
```

```
    G0_OPEN = 0,
```

```
};
```

```
enum Group1
```

```
{
```

```
    G1_CLOSE = 0,
```

```
    G1_OPEN = 1,
```

```
};
```

```
//Grammars and Words
```

```
enum Wordsets
```

```
{
```

```
    SET_1 = -1,
```

```
    SET_2 = -2,
```

```
SET_3 = -3,
```

```
};
```

```
enum Wordset1
```

```
{
```

```
S1_ACTION = 0,
```

```
S1_MOVE = 1,
```

```
S1_TURN = 2,
```

```
S1_RUN = 3,
```

```
S1_LOOK = 4,
```

```
S1_ATTACK = 5,
```

```
S1_STOP = 6,
```

```
S1_HELLO = 7,
```

```
};
```

```
enum Wordset2
```

```
{
```

```
S2_LEFT = 0,
```

```
S2_RIGHT = 1,
```

```
S2_UP = 2,
```

```
S2_DOWN = 3,
```

```
S2_FORWARD = 4,
```

```
S2_BACKWARD = 5,
```

```
};
```

```
enum Wordset3
```

```
{
```

```
S3_ZERO = 0,  
S3_ONE = 1,  
S3_TWO = 2,  
S3_THREE = 3,  
S3_FOUR = 4,  
S3_FIVE = 5,  
S3_SIX = 6,  
S3_SEVEN = 7,  
S3_EIGHT = 8,  
S3_NINE = 9,  
S3_TEN = 10,  
};
```

```
// use negative group for wordsets  
int8_t group, idx;
```

```
#include <Servo.h>
```

```
Servo myservo; // create servo object to control a servo
```

```
#define buz 9
```

```
void setup() {  
    // set up the LCD's number of columns and rows:  
  
    pinMode (buz,OUTPUT);
```

```
digitalWrite (buz,LOW);

lcd.begin(16, 2);
lcd.setCursor(0,0);
lcd.print("Brakas Elec.EsVR");
lcd.setCursor(0,1);
lcd.print("T.Elektro polsri");

myservo.attach(11); // attaches the servo on pin 9 to the servo object
myservo.write(90); // sets the servo position according to the scaled
value

// setup PC serial port
pcSerial.begin(9600);

bridge:
// bridge mode?
int mode = easyvr.bridgeRequested(pcSerial);
switch (mode)
{
case EasyVR::BRIDGE_NONE:
// setup EasyVR serial port
port.begin(9600);
// run normally
pcSerial.println(F("Bridge not requested, run normally"));
pcSerial.println(F("---"));
break;
```

```
case EasyVR::BRIDGE_NORMAL:  
    // setup EasyVR serial port (low speed)  
    port.begin(9600);  
  
    // soft-connect the two serial ports (PC and EasyVR)  
    easyvr.bridgeLoop(pcSerial);  
  
    // resume normally if aborted  
    pcSerial.println(F("Bridge connection aborted"));  
    pcSerial.println(F("---"));  
    break;  
  
case EasyVR::BRIDGE_BOOT:  
    // setup EasyVR serial port (high speed)  
    port.begin(115200);  
  
    pcSerial.end();  
    pcSerial.begin(115200);  
  
    // soft-connect the two serial ports (PC and EasyVR)  
    easyvr.bridgeLoop(pcSerial);  
  
    // resume normally if aborted  
    pcSerial.println(F("Bridge connection aborted"));  
    pcSerial.println(F("---"));  
    break;  
}  
  
// initialize EasyVR  
while (!easyvr.detect())  
{  
    pcSerial.println(F("EasyVR not detected!"));
```

```

for (int i = 0; i < 10; ++i)

{
    if (pcSerial.read() == '?')
        goto bridge;

    delay(100);

}

}

pcSerial.print(F("EasyVR detected, version "));

pcSerial.print(easyvr.getID());

if (easyvr.getID() < EasyVR::EASYVR3)
    easyvr.setPinOutput(EasyVR::IO1, LOW); // Shield 2.0 LED off

if (easyvr.getID() < EasyVR::EASYVR)
    pcSerial.print(F(" = VRbot module"));

else if (easyvr.getID() < EasyVR::EASYVR2)
    pcSerial.print(F(" = EasyVR module"));

else if (easyvr.getID() < EasyVR::EASYVR3)
    pcSerial.print(F(" = EasyVR 2 module"));

else
    pcSerial.print(F(" = EasyVR 3 module"));

    pcSerial.print(F(", FW Rev."));

    pcSerial.println(easyvr.getID() & 7);

easyvr.setDelay(0); // speed-up replies

```

```
easyvr.setTimeout(5);
easyvr.setLanguage(0);

group = EasyVR::TRIGGER; //<-- start group (customize)

// set up the LCD's number of columns and rows:
lcd.begin(16, 2);

Serial.begin(9600);

lcd.clear();

lcd.setCursor(0,0);
lcd.print("Silahkan Input");

lcd.setCursor(0,1);
lcd.print("Password");

}

int easy;

void loop(){

if(easy==1){
```

```

if (easyvr.getID() < EasyVR::EASYVR3)

    easyvr.setPinOutput(EasyVR::IO1, HIGH); // LED on (listening)

if (group < 0) // SI wordset/grammar

{

    pcSerial.print("Say a word in Wordset ");

    pcSerial.println(-group);

    easyvr.recognizeWord(-group);

}

else // SD group

{

    lcd.clear();

    delay(10);

    group=1;

    pcSerial.print("Say a command in Group ");

    pcSerial.println(group);

    easyvr.recognizeCommand(group);

    lcd.setCursor(0,0);

    lcd.print(" easy VR      ");

    lcd.setCursor(0,1);

    lcd.print(" Activ      ");

}

do

```

```

{

// allows Commander to request bridge on Zero (may interfere with user
protocol)

if (pcSerial.read() == '?')

{

setup();

return;

}

// <<-- can do some processing here, while the module is busy

}

while (!easyvr.hasFinished());


if (easyvr.getID() < EasyVR::EASYVR3)

easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off


idx = easyvr.getWord();

if (idx == 0 && group == EasyVR::TRIGGER)

{

// beep

easyvr.playSound(0, EasyVR::VOL_FULL);

// print debug message

pcSerial.println("Word: ROBOT");

// write your action code here

// group = GROUP_X\SET_X; <-- jump to another group or wordset

return;

}

else if (idx >= 0)

```

```
{  
    // beep  
    easyvr.playSound(0, EasyVR::VOL_FULL);  
  
    // print debug message  
    uint8_t flags = 0, num = 0;  
  
    char name[32];  
  
    pcSerial.print("Word: ");  
  
    pcSerial.print(idx);  
  
    if (easyvr.dumpGrammar(-group, flags, num))  
    {  
        for (uint8_t pos = 0; pos < num; ++pos)  
        {  
            if (!easyvr.getNextWordLabel(name))  
                break;  
  
            if (pos != idx)  
                continue;  
  
            pcSerial.print(F(" = "));  
            pcSerial.println(name);  
            break;  
        }  
    }  
  
    // perform some action  
    action();  
  
    return;  
}  
  
idx = easyvr.getCommand();  
if (idx >= 0)
```

```
{  
    // beep  
    easyvr.playSound(0, EasyVR::VOL_FULL);  
  
    // print debug message  
    uint8_t train = 0;  
  
    char name[32];  
  
    pcSerial.print("Command: ");  
    pcSerial.print(idx);  
  
    if (easyvr.dumpCommand(group, idx, name, train))  
    {  
        pcSerial.print(" = ");  
        pcSerial.println(name);  
    }  
  
    else  
        pcSerial.println();  
  
    // perform some action  
    action();  
}  
  
else // errors or timeout  
{  
    if (easyvr.isTimeout())  
        pcSerial.println("Timed out, try again...");  
  
    int16_t err = easyvr.getError();  
  
    if (err >= 0)  
    {  
        pcSerial.print("Error ");  
        pcSerial.println(err, HEX);  
    }  
}
```

```
    }

}

}

if(easy==0){

    char key = keypad.getKey();

    lcd.setCursor(0,0);
    lcd.print(" Password      ");

    lcd.setCursor(0,1);
    lcd.print("      ");

    if (key) {

        digit=digit+1;

        digitalWrite (buz,HIGH);

        lcd.setCursor(0,0);

        lcd.print(" Password      ");

        lcd.setCursor(0,1);

        lcd.print("      ");

    }

}
```

```
delay(50);

digitalWrite (buz,LOW);

if(key=='0'){pres=0;}
if(key=='1'){pres=1;}
if(key=='2'){pres=2;}
if(key=='3'){pres=3;}
if(key=='4'){pres=4;}
if(key=='5'){pres=5;}
if(key=='6'){pres=6;}
if(key=='7'){pres=7;}
if(key=='8'){pres=8;}
if(key=='9'){pres=9;}
```

```
delay(100);

if (digit==1){
    key1=pres*1000;
    lcd.setCursor(5,1);
    lcd.print("*");
}

if (digit==2){
    key2=pres*100;
```

```
lcd.setCursor(6,1);

lcd.print("*");

}

if (digit==3){

key3=pres*10;

lcd.setCursor(7,1);

lcd.print("*");

}

if (digit==4){

key4=pres*1;

lcd.setCursor(8,1);

lcd.print("*");

digit=5;

}

kode=key1+key2+key3+key4;

Serial.print("digit=");
```

```
Serial.println(digit);
Serial.print("key1=");
Serial.print(key1);
Serial.print("key2=");
Serial.print(key2);
Serial.print("key3=");
Serial.print(key3);
Serial.print("key4=");
Serial.println(key4);
Serial.println(key);
Serial.println(kode);

if (digit==5){

//=====

if(kode==1004)

//=====

{

lcd.setCursor(0,0);
lcd.print("KODE BERHASIL");

lcd.setCursor(0,1);
lcd.print("Voice Activation");
}
```

```
easy=1;

}

else

{



lcd.setCursor(0,0);
lcd.print("KODE DITOLAK");
digitalWrite (buz,HIGH);
delay(1000);

digitalWrite (buz,LOW);
delay(1000);
digitalWrite (buz,HIGH);
delay(1000);
digitalWrite (buz,LOW);
delay(1000);

digitalWrite (buz,HIGH);
delay(1000);
digitalWrite (buz,LOW);
delay(1000);

}

delay(3000);
```

```
lcd.clear();

lcd.setCursor(0,0);
lcd.print("Silahkan Input");

lcd.setCursor(0,1);
lcd.print("Password");

digit=0;
kode=0;

}

}

void action()
{
switch (group)
{
case GROUP_0:
switch (idx)
{
case G0_OPEN:
```

```

lcd.setCursor(0,0);
lcd.print("Perintah Open  ");
lcd.setCursor(0,1);
lcd.print(" diterima    ");

myservo.write(10);           // sets the servo position according to the scaled
value

delay(2000);

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

}

break;

case GROUP_1:

switch (idx)

{

case G1_CLOSE:

lcd.setCursor(0,0);
lcd.print("Perintah Close  ");
lcd.setCursor(0,1);
lcd.print(" diterima    ");

easy=0;

myservo.write(90);           // sets the servo position according to the scaled
value

```

```
delay(1000);

loop();

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

case G1_OPEN:

lcd.setCursor(0,0);
lcd.print("Perintah Open  ");
lcd.setCursor(0,1);
lcd.print(" diterima    ");

myservo.write(10);           // sets the servo position according to the scaled
value

delay(2000);

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

}

break;

case SET_1:
```

```
switch (idx)
{
    case S1_ACTION:
        // write your action code here
        // group = GROUP_X\SET_X; <-- or jump to another group or wordset for
        // composite commands
        break;

    case S1_MOVE:
        // write your action code here
        // group = GROUP_X\SET_X; <-- or jump to another group or wordset for
        // composite commands
        break;

    case S1_TURN:
        // write your action code here
        // group = GROUP_X\SET_X; <-- or jump to another group or wordset for
        // composite commands
        break;

    case S1_RUN:
        // write your action code here
        // group = GROUP_X\SET_X; <-- or jump to another group or wordset for
        // composite commands
        break;

    case S1_LOOK:
        // write your action code here
        // group = GROUP_X\SET_X; <-- or jump to another group or wordset for
        // composite commands
        break;

    case S1_ATTACK:
        // write your action code here
```

```
// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

case S1_STOP:

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

case S1_HELLO:

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

}

break;

case SET_2:

switch (idx)

{

case S2_LEFT:

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

case S2_RIGHT:

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

case S2_UP:
```

```

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

case S2_DOWN:

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

case S2_FORWARD:

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

case S2_BACKWARD:

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

}

break;

case SET_3:

switch (idx)

{

case S3_ZERO:

// write your action code here

// group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands

break;

```

```
case S3_ONE:  
    // write your action code here  
  
    // group = GROUP_X\SET_X; <-- or jump to another group or wordset for  
    // composite commands  
  
    break;  
  
case S3_TWO:  
    // write your action code here  
  
    // group = GROUP_X\SET_X; <-- or jump to another group or wordset for  
    // composite commands  
  
    break;  
  
case S3_THREE:  
    // write your action code here  
  
    // group = GROUP_X\SET_X; <-- or jump to another group or wordset for  
    // composite commands  
  
    break;  
  
case S3_FOUR:  
    // write your action code here  
  
    // group = GROUP_X\SET_X; <-- or jump to another group or wordset for  
    // composite commands  
  
    break;  
  
case S3_FIVE:  
    // write your action code here  
  
    // group = GROUP_X\SET_X; <-- or jump to another group or wordset for  
    // composite commands  
  
    break;  
  
case S3_SIX:  
    // write your action code here  
  
    // group = GROUP_X\SET_X; <-- or jump to another group or wordset for  
    // composite commands
```

```
break;

case S3_SEVEN:
    // write your action code here
    // group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands
    break;

case S3_EIGHT:
    // write your action code here
    // group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands
    break;

case S3_NINE:
    // write your action code here
    // group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands
    break;

case S3_TEN:
    // write your action code here
    // group = GROUP_X\SET_X; <-- or jump to another group or wordset for
composite commands
    break;
}

break;
}
```