LAMPIRAN 1

1. Skema Rangkaian Arduino dengan Piezoelektrik
2. Rangkaian Skema Arduino dengan Tombol Rekam

![Diagram Arduino dengan Tombol Rekam]

3. Rangkaian Skema Arduino dengan Solenoid DC

![Diagram Arduino dengan Solenoid DC]
LAMPIRAN 2
Coding Pintu Otomatis Menggunakan Pola Ketukan

Analog Pin 0: Piezo speaker
Digital Pin 2: Digunakan untuk memprogram ketukan
Digital Pin 3: Relay
Digital Pin 4: Red LED.
Digital Pin 5: Green LED.
Digital Pin 6: Buzzer
*/

#define RELAY_ON 0
#define RELAY_OFF 1
#define RELAY_1 3 // pin yang digunakan yaitu pin 3

// Pin yang digunakan
const int knockSensor = 0;    // Piezo sensor di pin 0.
const int programSwitch = 2; // digunakan untuk memprogram ketukan baru
const int redLED = 4;         // Status LED
const int greenLED = 5;       // Status LED
const int buzzer = 6;

const int threshold = 3;      // Minimum signal from the piezo to register as a knock
const int rejectValue = 25;   // If an individual knock is off by this percentage of a knock we don't unlock..
const int averageRejectValue = 15; // If the average timing of the knocks is off by this percent we don't unlock.
const int knockFadeTime = 150;  // milliseconds we allow a knock to fade before we listen for another one. (Debounce timer.)
const int lockTurnTime = 650;   // milliseconds that we run the motor to get it to go a half turn.
const int maximumKnocks = 20;   // Maximum number of knocks to listen for.
const int knockComplete = 1200; // Longest time to wait for a knock before we assume that it's finished.

// Variables.
int secretCode[maximumKnocks] = {50, 25, 25, 50, 100, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int knockReadings[maximumKnocks];
int knockSensorValue = 0;
int programButtonPressed = false;

void setup() {
    pinMode(RELAY_1, OUTPUT);
digitalWrite(RELAY_1, RELAY_OFF);
    pinMode(redLED, OUTPUT);
    pinMode(greenLED, OUTPUT);
    pinMode(buzzer, OUTPUT);
    pinMode(programSwitch, INPUT);

    Serial.begin(9600);
    Serial.println("Program start.");
digitalWrite(greenLED, HIGH);
}
void loop() {
  // Listen for any knock at all.
  knockSensorValue = analogRead(knockSensor);

  if (digitalRead(programSwitch)==HIGH){ // is the program button pressed?
    programButtonPressed = true;    // Yes, so lets save that state
    digitalWrite(redLED, HIGH);      // and turn on the red light too so we know
    we're programming.
  } else {
    programButtonPressed = false;
    digitalWrite(redLED, LOW);
  }

  if (knockSensorValue >=threshold){
    listenToSecretKnock();
  }
}

// Records the timing of knocks.
void listenToSecretKnock(){
  Serial.println("knock starting");
  int i = 0;
  // First lets reset the listening array.
  for (i=0;i<maximumKnocks;i++){
    knockReadings[i]=0;
  }
}
int currentKnockNumber=0; // Incrementer for the array.
int startTime=millis(); // Reference for when this knock started.
int now=millis();
digitalWrite(greenLED, LOW); // we blink the LED for a bit as a visual indicator of the knock.
if (programButtonPressed==true){
digitalWrite(redLED, LOW); // and the red one too if we're programming a new knock.
}
delay(knockFadeTime); // wait for this peak to fade before we listen to the next one.
digitalWrite(greenLED, HIGH);
if (programButtonPressed==true){
digitalWrite(redLED, HIGH);
}
do {
//listen for the next knock or wait for it to timeout.
knockSensorValue = analogRead(knockSensor);
if (knockSensorValue >=threshold){
//record the delay time.
Serial.println("knock.");
now=millis();
knockReadings[currentKnockNumber] = now-startTime;
currentKnockNumber ++;
startTime=now;
// and reset our timer for the next knock
digitalWrite(greenLED, LOW);
if (programButtonPressed==true) {
    digitalWrite(redLED, LOW);  // and the red one too if we're programming a new knock.
}

delay(knockFadeTime);  // again, a little delay to let the knock decay.
digitalWrite(greenLED, HIGH);
if (programButtonPressed==true) {
    digitalWrite(redLED, HIGH);
}
}

now=millis();

// did we timeout or run out of knocks?
} while ((now-startTime < knockComplete) && (currentKnockNumber < maximumKnocks));

// we've got our knock recorded, lets see if it's valid
if (programButtonPressed==false) {  // only if we're not in progrmaing mode.
    if (validateKnock() == true) {
        triggerDoorUnlock();
    } else {
        Serial.println("Secret knock failed.");
        digitalWrite(greenLED, LOW);  // We didn't unlock, so blink the red LED as visual feedback.
for (i=0;i<4;i++){
    digitalWrite(redLED, HIGH);
    digitalWrite(buzzer, HIGH);
    delay(1000);
    digitalWrite(redLED, LOW);
    digitalWrite(buzzer, LOW);
    delay(100);
}
digitalWrite(greenLED, HIGH);
}
} else { // if we're in programming mode we still validate the lock, we just don't do anything with the lock
    validateKnock();
    // and we blink the green and red alternately to show that program is complete.
    Serial.println("New lock stored.");
    digitalWrite(redLED, LOW);
    digitalWrite(greenLED, LOW);
    digitalWrite(greenLED, HIGH);
    for (i=0;i<3;i++){
        delay(100);
        digitalWrite(redLED, HIGH);
        digitalWrite(greenLED, LOW);
        delay(100);
        digitalWrite(redLED, LOW);
        digitalWrite(greenLED, LOW);
        digitalWrite(greenLED, HIGH);
    }
}
void triggerDoorUnlock() {
    Serial.println("Door unlocked!");
    int i=0;

    digitalWrite(RELAY_1, RELAY_ON);   // lampu menyala 2 detik
    digitalWrite(greenLED, HIGH);      // And the green LED too.
    delay (10000);                      // Wait a bit.
    digitalWrite(RELAY_1, RELAY_OFF);

    // Blink the green LED a few times for more visual feedback.
    for (i=0; i < 5; i++){
        digitalWrite(greenLED, LOW);
        delay(100);
        digitalWrite(greenLED, HIGH);
        delay(100);
    }
}

// Sees if our knock matches the secret.
// returns true if it's a good knock, false if it's not.
// todo: break it into smaller functions for readability.
boolean validateKnock() {
    int i=0;

    // simplest check first: Did we get the right number of knocks?
int currentKnockCount = 0;
int secretKnockCount = 0;
int maxKnockInterval = 0;       // We use this later to normalize the times.

for (i=0;i<maximumKnocks;i++){
    if (knockReadings[i] > 0){
        currentKnockCount++;
    }
    if (secretCode[i] > 0){    //todo: precalculate this.
        secretKnockCount++;
    }
}

if (knockReadings[i] > maxKnockInterval){   // collect normalization data while we're looping.
    maxKnockInterval = knockReadings[i];
}

// If we're recording a new knock, save the info and get out of here.
if (programButtonPressed==true){
    for (i=0;i<maximumKnocks;i++){    // normalize the times
        secretCode[i] = map(knockReadings[i],0,maxKnockInterval,0,100);
    }
    // And flash the lights in the recorded pattern to let us know it's been programmed.
    digitalWrite(greenLED, LOW);
}
digitalWrite(redLED, LOW);
delay(1000);
digitalWrite(greenLED, HIGH);
digitalWrite(redLED, HIGH);
delay(50);
for (i = 0; i < maximumKnocks ; i++){
digitalWrite(greenLED, LOW);
digitalWrite(redLED, LOW);
// only turn it on if there's a delay
if (secretCode[i] > 0){
delay( map(secretCode[i],0, 100, 0, maxKnockInterval)); // Expand the time back out to what it was. Roughly.
digitalWrite(greenLED, HIGH);
digitalWrite(redLED, HIGH);
}
delay(50);
}
return false;  // We don't unlock the door when we are recording a new knock.
}

if (currentKnockCount != secretKnockCount){
 return false;
}

/*/ untuk membandingkan interval relatif mengetuk pada sensor */
int totaltimeDifferences=0;
int timeDiff=0;
for (i=0;i<maximumKnocks;i++){
    knockReadings[i]= map(knockReadings[i],0, maxKnockInterval, 0, 100);
    timeDiff = abs(knockReadings[i]-secretCode[i]);
    if (timeDiff > rejectValue){
        return false;
    }
    totaltimeDifferences += timeDiff;
}
if (totaltimeDifferences/secretKnockCount>averageRejectValue){
    return false;
}
return true;