Sarah Wood

Embedded Programming

Chapter 11

Assignment 2

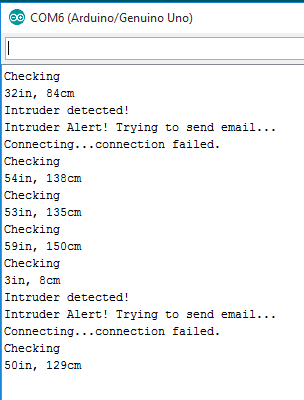
Build a project similar to the burglar alarm, but use another type of sensor.

There’s tons of inspiration out there on the Web.9

I used the ultrasonic ranger. I could stick it to my front door. This would be far more pet-friendly than Passive IR. The wall opposing my front door is several yards away, whereas when the door opens, the nearest wall is about a yard.

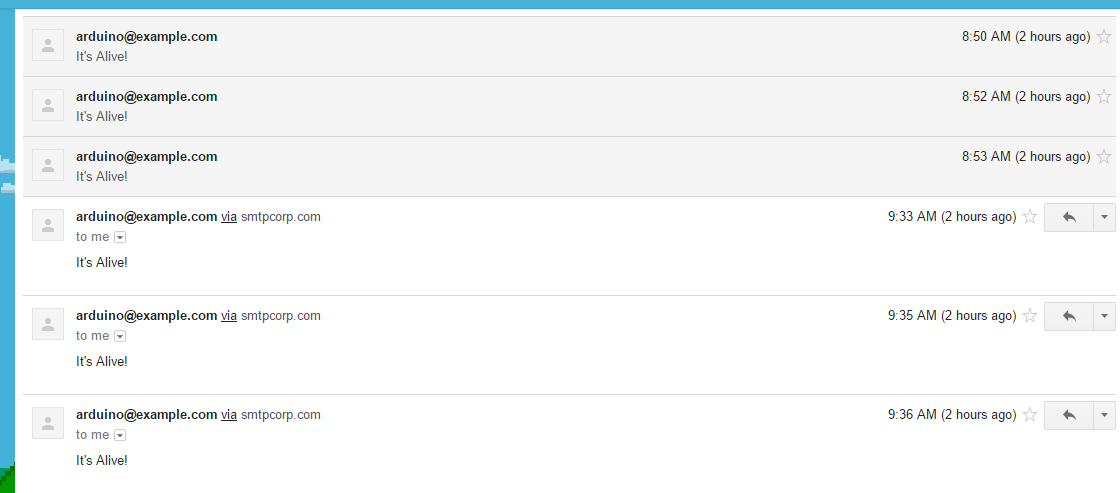
I had time to test it in class.

Screen Caps



Hardware interface is OK. Now to steal Ethernet from this machine for a minute.





The emails from the last project are greyed out. The three sent around 9:35 are the ones sent from the ultrasonic ranger.

Source Code

(Only main & pir\_sensor.h were changed)

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

Main.cpp

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

#include <SPI.h>

#include <Ethernet.h>

#include "burglar\_alarm.h"

const unsigned int INPUT\_PIN = 7;

const unsigned int SMTP\_PORT = 2525; // as per SMTP2go

const unsigned int BAUD\_RATE = 9600;

const String USERNAME = "eWFybmJvbWJlckBnbWFpbC5jb20=="; // Encoded in Base64.

const String PASSWORD = "MkJlU2JWVWZNUlpzDQo=="; // Encoded in Base64.

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

IPAddress my\_ip(205, 122, 107, 29);

// Insert IP address of your SMTP server below!

IPAddress smtp\_server(216, 22, 15, 250); // inserted SMTP2go static IP

Sensor sensor(INPUT\_PIN);

SmtpService smtp\_service(smtp\_server, SMTP\_PORT, USERNAME, PASSWORD);

BurglarAlarm burglar\_alarm(sensor, smtp\_service);

void setup() {

Ethernet.begin(mac, my\_ip);

Serial.begin(BAUD\_RATE);

delay(20 \* 1000);

}

void loop() {

burglar\_alarm.check();

delay(3000);

}

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

pir\_sensor.h (I know it’s not actually a PIR)

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

#ifndef \_sensor\_h

#define \_sensor\_h

class Sensor {

int \_input\_pin;

int \_trigger\_pin;

public:

Sensor(const int input\_pin) {

\_input\_pin = input\_pin;

pinMode(\_input\_pin, INPUT);

}

const bool motion\_detected() const {

//this is where I need to run the ranger to return motion true or !true

// establish variables for duration of the ping,

// and the distance result in inches and centimeters:

long duration, inches, cm;

const int pingPin = 7;

const int triggerPin = 6;

// The PING))) is triggered by a HIGH pulse of 2 or more microseconds.

// Give a short LOW pulse beforehand to ensure a clean HIGH pulse:

pinMode(triggerPin, OUTPUT);

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

digitalWrite(triggerPin, HIGH);

delayMicroseconds(5);

digitalWrite(triggerPin, LOW);

// The same pin is used to read the signal from the PING))): a HIGH

// pulse whose duration is the time (in microseconds) from the sending

// of the ping to the reception of its echo off of an object.

pinMode(pingPin, INPUT);

duration = pulseIn(pingPin, HIGH);

// convert the time into a distance

inches = (duration / 74 / 2);

cm = (duration / 29 / 2);

Serial.print(inches);

Serial.print("in, ");

Serial.print(cm);

Serial.print("cm");

Serial.println();

delay(100);

if (inches <50)

{return true;}

else

{return false;}

//return digitalRead(\_input\_pin) == HIGH; //legacy code from PIR sensor

}

};

#endif