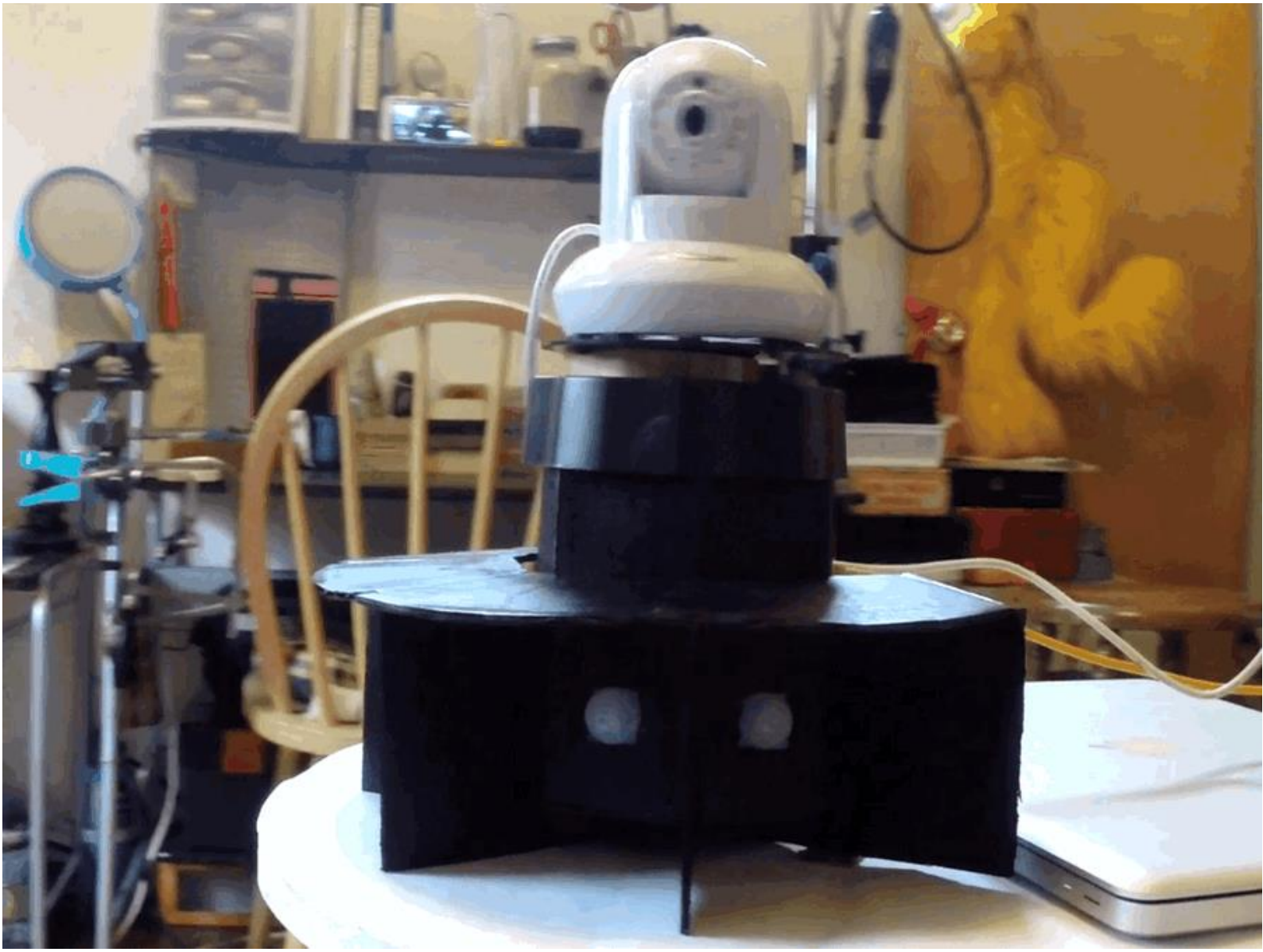
Base da Câmera de Movimento Motorizado



COISAS UTILIZADAS NESTE PROJETO

Componentes de hardware:

	Arduino UNO &Genuino UNO	× 1
	Capacitor 100 µF	× 1
	LED (generic)	× 5
	PIR Motion Sensor (generic)	× 5
	Resistor, 220 ohm	× 5
	Servo (generic)	× 1

Relato

Eu vim acima com esta ideia para resolver um problema que eu estava tendo com minhas câmeras de segurança em casa. Eu tenho uma câmera particular que está posicionada em uma parede entre dois quartos, no entanto, você só pode ver um quarto de cada vez, a menos que você login para o software da câmera para girá-lo manualmente. Se alguém fosse entrar no quarto oposto, a câmera nunca gravaria a ação. Para resolver este problema, eu decidi criar uma base de detecção / rastreamento de movimento na qual eu poderia anexar a câmera para que ela se reposicionasse automaticamente para onde quer que o movimento fosse detectado.

COMO FUNCIONA:

Este dispositivo utiliza 5 sensores de movimento PIR para detectar qualquer movimento dentro de um raio de 180º. Uma vez que o movimento é detectado, um servo motor gira a base da câmera para apontar nessa direção. Há também 5 LED "status" luzes posicionadas dentro da base que vai acender a qualquer momento um dos sensores PIR detecta movimento.

FIAÇÃO:

A fiação para este projeto é simples e direta, no entanto, devido ao fato de que há 5 ao todo, exceto o servo motor, o número de fios começa a aumentar um pouco. Confira o diagrama de circuito e esquemas na parte inferior desta página para obter mais detalhes, mas aqui estão os conceitos básicos.

As luzes de status LED ...

Eu usei um mini breadboard para manter as luzes LED separadas do resto do circuito, a fim de permitir a remoção mais fácil, se necessário. Cada LED responde a um sensor PIR diferente e acende-se sempre que detecta movimento. Basta conectar cada LED a uma entrada digital separada (9-13) no Arduino, e aterrá-los através de um resistor de 220 ohms.

Os sensores de movimento PIR ...

Eu usei os sensores HC-SR501 modelo PIR para este projeto. Cada sensor PIR precisa ser conectado a uma entrada digital (2-6) no Arduino, alimentação de 5V e terra. Se você estiver usando o mesmo sensor PIR como eu, você pode (com cuidado) estourar a tampa e os pinos estão marcados embaixo. Você conectará cada sensor PIR à luz LED correspondente no código enviado posteriormente.

O servo motor ...

O servo motor é conectado ao suporte onde a câmera se senta e gira a câmera para alinhar com o sensor PIR ativo. Usando o pinout para o seu motor, ligue um pino à alimentação, outro pino à entrada digital Arduino 7

e o último pino à terra. Antes de alimentar o Arduino, certifique-se de conectar um capacitor de 100µF entre a energia e o motor para ajudar a proteger a placa contra os surtos de energia que ocorrem quando o motor se move.

Uma coisa a notar sobre os servos motores é que nem todos têm uma gama completa de 180º de movimento. Depois de alguns testes, descobri que o meu só move cerca de 160 º, de modo a modificar o código em conformidade, se o seu motor difere. Você saberá que o motor está tentando mover-se demasiado distante quando faz um ruído de moedura na última posição do sensor de PIR.

CONSTRUÇÃO:

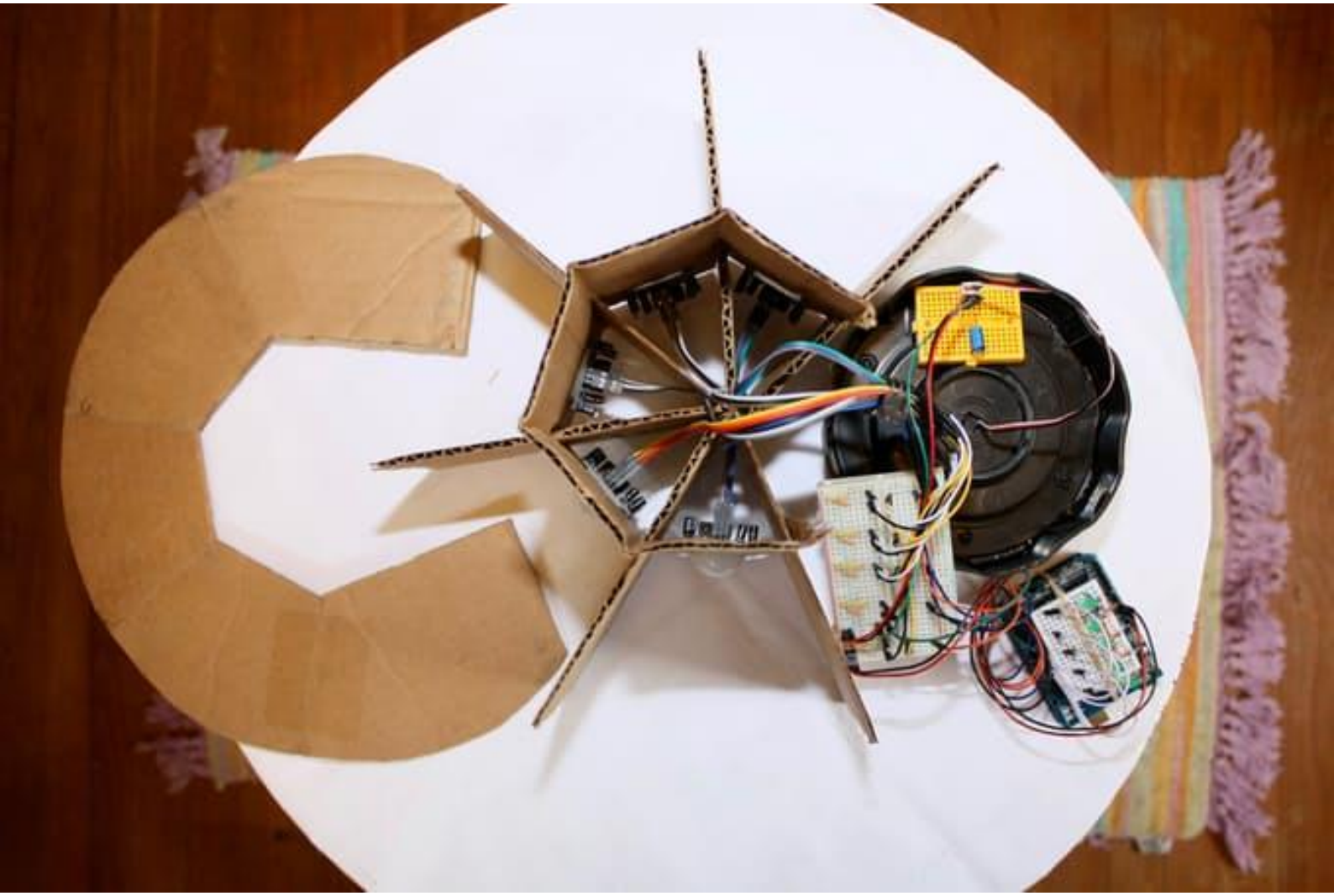
Eu usei um rótulo velho de Memorex cd para a carcaça do servo e a superfície para que uma câmera se sente sobre ele. Eu não consegui tirar uma foto 'antes', então a imagem abaixo é a única restante que eu poderia encontrar on-line. A parte de trás era um disco de plástico plano, resistente que eventualmente saiu em uma peça (depois de um pouco de persuasão da minha chave de fenda), de modo que funcionou muito bem para sentar a minha câmera. Ele também veio com 4 pés de borracha removível, o que ajuda a dar a câmera um pouco mais aderência quando o motor se move.

Eu levei meus alicates para a parte preta no centro para removê-lo ... e por isso quero dizer, quebrá-lo em tantas partes quanto eu puder antes que ele tenha ido :) Em seguida, eu perfurei dois buracos na base com a minha Dremel e, em seguida, Liguei o servo motor com um par de parafusos.

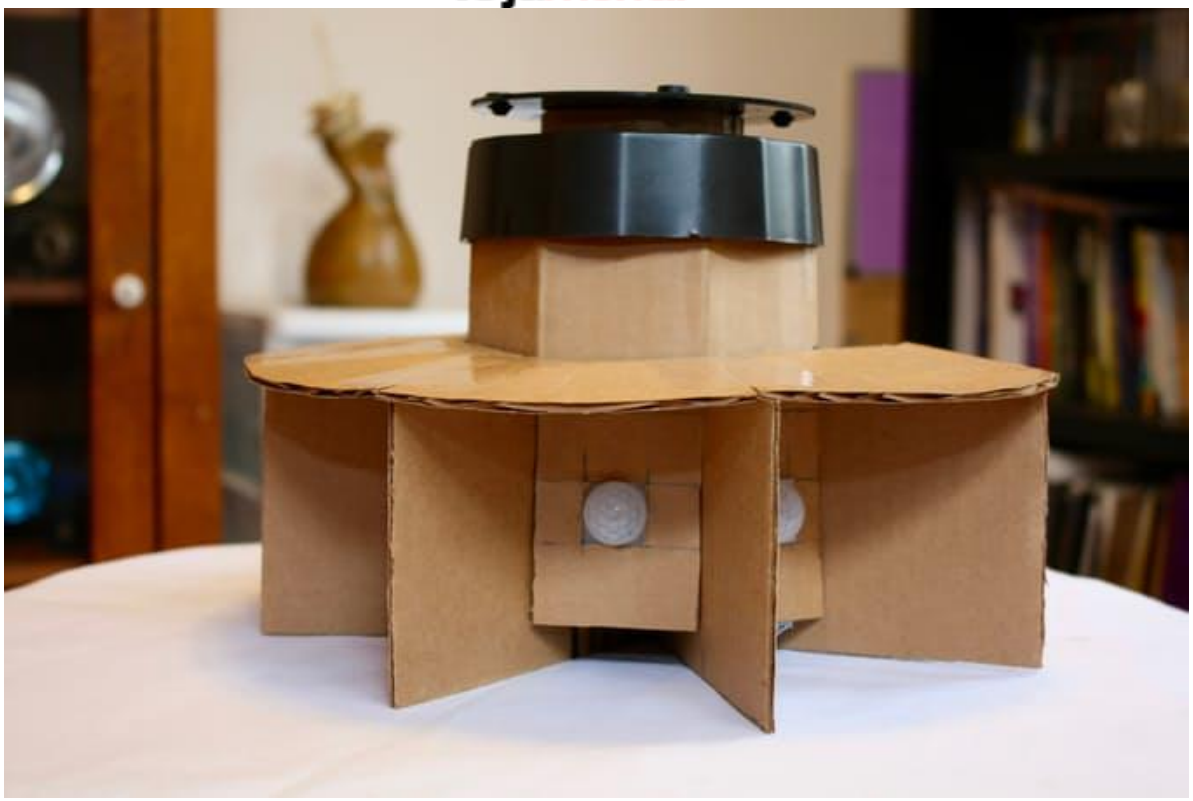
Por último, eu usei um outro parafuso para anexar o servo motor a peça onde a câmera vai se sentar. Eu tive um problema o peso da câmera estava fazendo com que a base se inclinasse, então eu resolvi isso, fazendo um cilindro de papelão alto o suficiente para caber entre o motor e a câmera. Agora ele se move livremente, mas tem mais apoio com seu peso uniformemente distribuído.

O servo motor está ligado à base do etiquetador de CD e a parte inferior é reutilizada como superfície para sentar a câmera. O cartão circular é adicionado entre as duas peças para a estabilidade.

Aqui está tudo o que aparece depois que os sensores PIR foram instalados e todos os circuitos concluídos. Você notará que eu usei vários painéis para completar meu projeto. A razão para isso é que ele tornou extremamente fácil de colocá-los dentro da base, bem como escolher qual remover durante a resolução de problemas e expandir mais tarde.



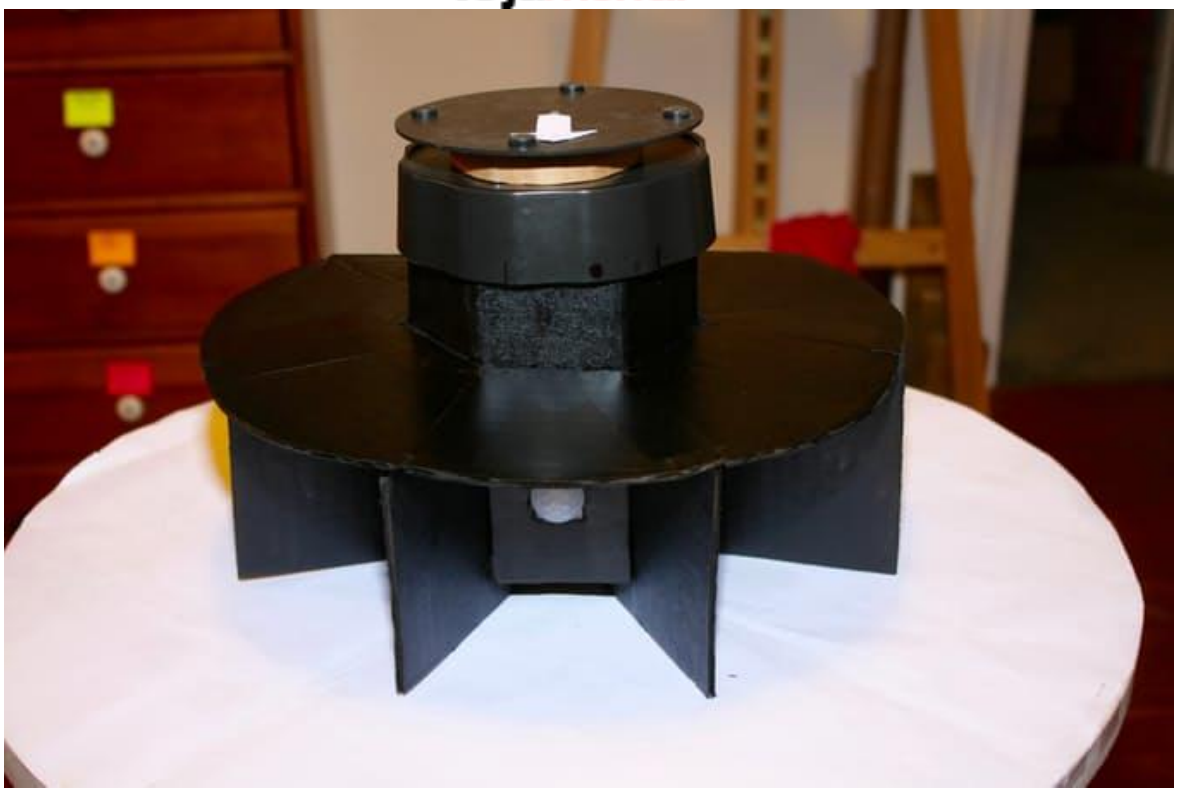
A base não pintada e suporte com toda a fiação concluída

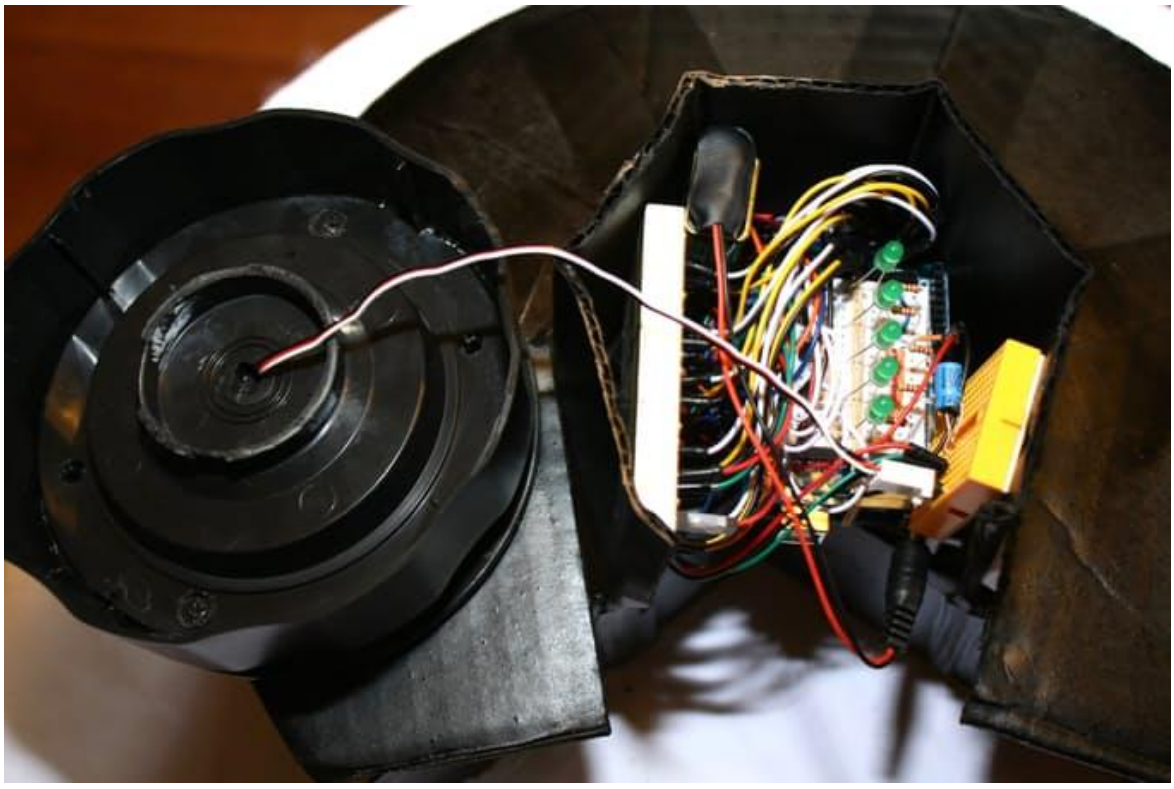


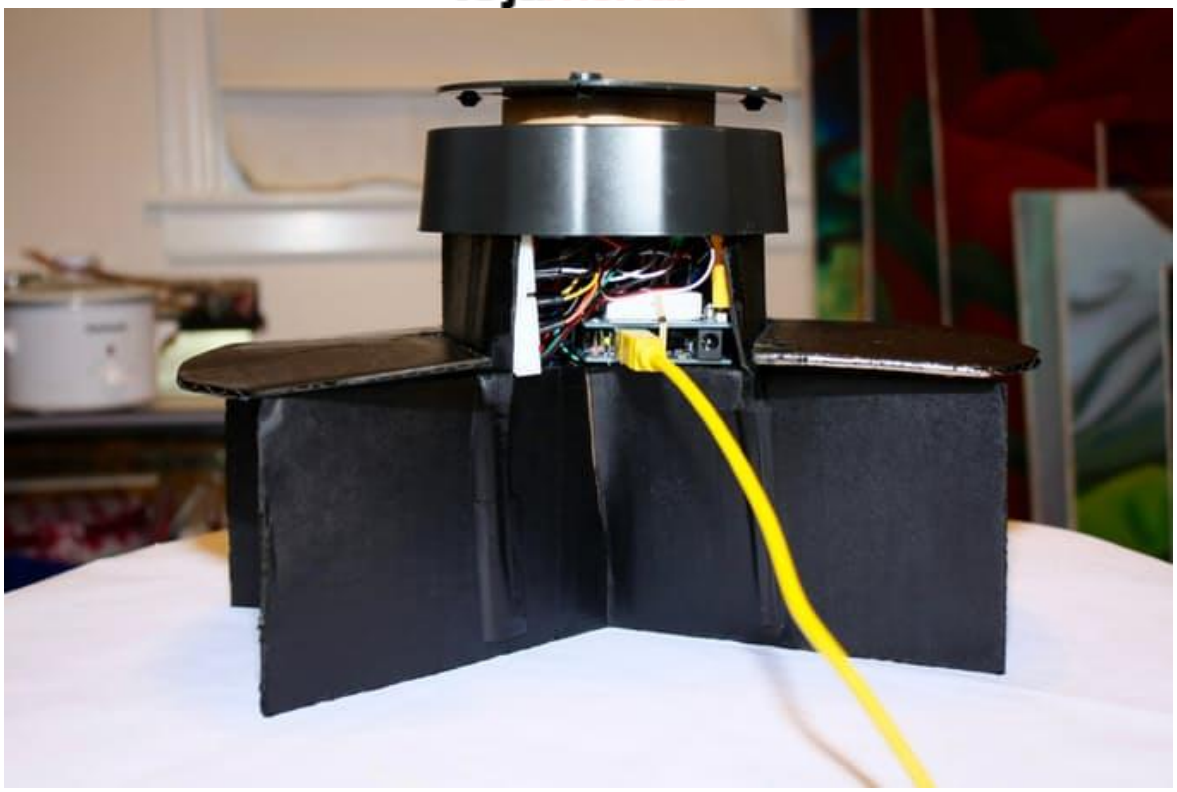
A base não pintada no final / stand com toda a fiação escondida logo abaixo da base

Para completar o projeto, eu pintei com spray preto as partes visíveis do papelão para que ele pareça mais transparente com o suporte preto acima.

Aqui está o produto acabado depois de pintar ...







SCHEMATICS

Circuit Diagram

Download

Schematics

Download

Base/Stand Construction - Part 1

Download

Base/Stand Construction - Part 2

Download

CODE

The Main Code**Arduino**

The PIR Sensors will calibrate for 15 seconds after powering up the Arduino. It is recommended that you leave the room during this process so that the sensors get a true image of the room without any motion.

```
/*************************************************
```

```
*** The Motion Following Motorized Camera Base ***
***                by Lindsay Fox               ***
************************************************/

// Servor motor
#include<Servo.h>
ServocamServo;// name the servo motor controlling the camera base
intcurrentPIRposition=0;// set current angle of servo

// LED status lights
intLEDpin[]={9,10,11,12,13};// LED pin numbers
intcurrentLEDpin=9;// the current LED pin; begin with the first in the
sequence above

// PIR sensors
intPIRpin[]={2,3,4,5,6};// PIR pin numbers
intcurrentPIRpin=2;// the current PIR pin; begin with the first in the
sequence above
intPIRprevState[]={1,1,1,1,1};// the previous state of the PIR (0 = LOW, 1 =
HIGH)
intPIRposition[]={157,117.75,78.5,39.25,0};// assign angles for servo motor
(0-157 distributed equally between 5 PIR sensors)
booleanPIRstatus;// Set status of PIR sensor as either true or false



///// SETUP //////////////////////////////////
voidsetup(){

Serial.begin(9600);
camServo.attach(7);// assign servo pin

for(intp=0;p<5;p++){// set all PIR sensors as INPUTS
pinMode(PIRpin[p],INPUT);
}// end 'p' for

for(intl=0;l<5;l++){// set all LEDs as OUTPUTS
pinMode(LEDpin[l],OUTPUT);
}// end 'l' for

/////// CALIBRATE PIR SENSORS ///////
Serial.print("Calibrating PIR Sensors ");
for(intc=0;c<15;c++){// calibrate PIR sensors for 15 seconds (change from 10-
60 sec depending on your sensors)
Serial.print(".");
delay(1000);// wait 1 second
}// end calibration for
Serial.println("PIR Sensors Ready");

camServo.write(78.5);// move the servo to the center position to begin

}// end setup
```

```
///// MAIN LOOP ////////////////////////////////
voidloop(){

for(intPIR=0;PIR<5;PIR++){// start this loop for each PIR sensor
currentPIRpin=PIRpin[PIR];// set current PIR pin to current number in 'for'
loop
currentLEDpin=LEDpin[PIR];// set current LED pin to current number in 'for'
loop
PIRstatus=digitalRead(currentPIRpin);

if(PIRstatus==HIGH){// if motion is detected on current PIR sensor
digitalWrite(currentLEDpin,HIGH);// turn corresponding LED on
if(PIRprevState[PIR]==0){// if PIR sensor's previous state is LOW
if(currentPIRposition!=currentPIRpin&&PIRprevState[PIR]==0){// if high PIR is
different than current position PIR then move to new position
camServo.write(PIRposition[PIR]);
Serial.print("Current angle : ");
Serial.println(PIRposition[PIR]);
delay(50);
currentPIRposition=currentPIRpin;// reset current PIR position to active
[PIR] pin
PIRprevState[PIR]=1;// set previous PIR state to HIGH
}
PIRprevState[PIR]=1;// set previous PIR state to HIGH if the current position
is the same as the current PIR pin
}// end PIRprevState if
}// end PIRstatus if

else{//
digitalWrite(currentLEDpin,LOW);//the led visualizes the sensors output pin
state
PIRprevState[PIR]=0;// set previous PIR state to LOW
}// end else

}// end [PIR] for loop
}// endmain loop
```

Anterior Próximo
1
2
3

SCHEMATICS
Diagrama de circuito
Transferir
Esquemas
Transferir
Construção de Base / Stand - Parte 1
Transferir
Construção de Base / Stand - Parte 2
Transferir

CÓDIGO
O código principalArduino
Os Sensores PIR serão calibrados por 15 segundos depois de ligar o Arduino.
Recomenda-se que você deixe a sala durante este processo para que os sensores
obter uma imagem verdadeira da sala sem qualquer movimento.

```
/ ***********************************************
*** O movimento que segue a base da câmera motorizada ***
*** por Lindsay Fox ***
********************************************** /

// Servor motor
#include <Servo.h>
ServocamServo // nome do servo motor que controla a base da câmera
IntcurrentPIRposition = 0; // define o ângulo atual do servo

// Luzes LED de status
IntLEDpin [] = {9,10,11,12,13}; // Número de pinos LED
IntcurrentLEDpin = 9; // o atual pino LED; Começar com o primeiro na
seqüência acima

// Sensores PIR
IntPIRpin [] = {2,3,4,5,6}; // PIR números de pinos
IntcurrentPIRpin = 2; // o pino PIR atual; Começar com o primeiro na
seqüência acima
IntPIRprevState [] = {1,1,1,1,1}; // o estado anterior do PIR (0 = LOW, 1 =
HIGH)
IntPIRposition [] = {157,117,75,78,5,39,25,0}; // atribuir ângulos para servo
motor (0-157 distribuídos igualmente entre 5 sensores PIR)
BooleanPIRstatus; // Definir o status do sensor PIR como verdadeiro ou falso



///// CONFIGURAÇÃO ////////////////////////////////////
Voidsetup () {

Serial.begin (9600);
CamServo.attach (7) // atribuir pino servo

Para (intp = 0; p <5; p ++) {// define todos os sensores PIR como ENTRADAS
PinMode (PIRpin [p], INPUT);
} // end 'p' para

Para (intl = 0; l <5; l ++) {// define todos os LEDs como OUTPUTS
PinMode (LEDpin [l], OUTPUT);
} // end 'l' para

/////// CALIBRAR PIR SENSORES ///////
Serial.print ("Calibrar Sensores PIR");
Para (intc = 0; c <15; c ++) {// calibre os sensores PIR por 15 segundos
(mude de 10-60 segundos dependendo de seus sensores)
Serial.print (".");
Atraso (1000); // espera 1 segundo
} // fim da calibração para
```

```
Serial.println ("Sensores PIR Pronto");

CamServo.write (78.5); // move o servo para a posição central para começar

} // fim da configuração



///// BOTÃO PRINCIPAL ////////////////////////////////
Voidloop () {

Para (intPIR = 0; PIR <5; PIR ++) {// iniciar este loop para cada sensor PIR
CurrentPIRpin = PIRpin [PIR]; // define o pino PIR atual para o número atual
no loop 'for'
CurrentLEDpin = LEDpin [PIR]; // define o pino do LED atual para o número
atual no loop 'for'
PIRstatus = digitalRead (currentPIRpin);

If (PIRstatus == HIGH) {// se for detectado movimento no sensor PIR atual
DigitalWrite (currentLEDpin, HIGH); // acende o LED correspondente
If (PIRprevState [PIR] == 0) {// se o estado anterior do sensor PIR for LOW
If (currentPIRposition! = CurrentPIRpin && PIRprevState [PIR] == 0) {// se o
PIR alto for diferente da posição atual PIR então mova para a nova posição
CamServo.write (PIRposition [PIR]);
Serial.print ("Ângulo atual:");
Serial.println (PIRposition [PIR]);
Atraso (50);
CurrentPIRposition = currentPIRpin; // redefine a posição atual do PIR para o
pino [PIR] ativo
PIRprevState [PIR] = 1; // definir o estado PIR anterior para HIGH
}
PIRprevState [PIR] = 1; // define o estado PIR anterior para HIGH se a
posição atual for igual ao pino PIR atual
} // end PIRprevState if
} // fim PIRstatus se

outro{//
DigitalWrite (currentLEDpin, LOW); // o led visualiza o estado do pino de
saída dos sensores
PIRprevState [PIR] = 0; // define o estado PIR anterior como LOW
} // end else

} // end [PIR] for loop
} // endmain loop
```