**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

**BỘ MÔN ĐIỆN TỬ**

---------------o0o---------------

****

**ĐỒ ÁN MÔN HỌC**

**GIẢI MÃ TÍN HIỆU HỒNG NGOẠI  
 CỦA REMOTE TV SONY**

**GVHD: Nguyễn Trọng Luật**

**SVTH: Phùng Đức Nghĩa**

**MSSV: 1412474**

**TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2017**

***LỜI CẢM ƠN***

Em xin gửi lời cảm ơn đến Bộ môn Điện tử - Viễn thông khoa Điện – Điên tử đã tạo điều kiện để em thực hiện đồ án này.

Để hoàn thành đồ án này, em xin gửi lời cảm ơn đến Ths. Nguyễn Trọng Luật đã tận tình hướng dẫn, giảng dạy em trong suốt quá trình học tập, nghiên cứu và rèn luyện để có thể thực hiện tốt đồ án này.

Mặc dù đã rất cố gắng để thực hiện bài tập lớn một cách tốt nhất. Song do sự hạn chế về kiến thức chuyên môn cũng như kinh nghiệm thực tế nên không tránh khỏi những thiếu sót. Vì vậy em rất mong được sự góp ý của quý thầy cô và các bạn để đề tài này được hoàn chỉnh hơn.

Em xin chân thành cảm ơn!

*Tp. Hồ Chí Minh, ngày 10 tháng 12 năm 2017 .*

**Sinh viên**

**TÓM TẮT ĐỒ ÁN**

Điều khiển bằng hồng ngoại hiện nay được sử dụng rất nhiều trong cuộc sống, ngoài việc điều khiển các thiết bị chuyên dụng như TV, quạt, máy lạnh, hồng ngoại còn được ứng dụng trong điều khiển thông minh trong nhà.

Đồ án này trình bày về cách thức giải mã tín hiệu từ remote TV Sony, một phần trong các ứng điều khiển thông minh bằng hồng ngoại.

**MỤC LỤC**

[1. GIỚI THIỆU 1](#_Toc502179046)

[1.1 Tổng quan 1](#_Toc502179047)

[1.2 Nhiệm vụ đề tài 1](#_Toc502179048)

[2. LÝ THUYẾT 1](#_Toc502179049)

[2.1 Tìm hiểu về tia hồng ngoại của remote TV 1](#_Toc502179050)

[2.2 Tìm hiểu về LED thu hồng ngoại TSOP1838 2](#_Toc502179051)

[2.3 Tìm hiểu về tín hiệu hồng ngoại từ remote TV Sony 3](#_Toc502179052)

[3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG 4](#_Toc502179053)

[4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM 8](#_Toc502179054)

[5. KẾT QUẢ THỰC HIỆN 18](#_Toc502179055)

[6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 18](#_Toc502179056)

[6.1 Kết luận 18](#_Toc502179057)

[6.2 Hướng phát triển 19](#_Toc502179058)

[7. TÀI LIỆU THAM KHẢO 19](#_Toc502179059)

[8. PHỤ LỤC 19](#_Toc502179060)

DANH SÁCH HÌNH MINH HỌA

[Hình 2.1 Ánh sáng hồng ngoại được điều chế và giải điều chế 2](#_Toc502185039)

[Hình 2.2 TSOP1838 2](file:///D:\Project\IR_Decode\document\bao_cao1.docx#_Toc502185040)

[Hình 2.3 Tín hiệu hồng ngoại thu được khi ấn nút “1” của remote TV Sony 3](#_Toc502185041)

[Hình 2.4 Tín hiệu hồng ngoại thu được khi ấn nút “2” của remote TV Sony 3](#_Toc502185042)

[Hình 2.5 Tín hiệu hồng ngoại thu được khi ấn nút Power của remote TV Sony 3](#_Toc502185043)

[Hình 2.6 Xung tín hiệu 1 và tín hiệu 0 của remote TV Sony. Nguồn: http://www.sbprojects.com/knowledge/ir/ 4](#_Toc502185044)

[Hình 2.7 Tổng quát 1 chuỗi xung nhận được từ remote TV Sony. Nguồn: http://www.sbprojects.com/knowledge/ir/ 4](#_Toc502185045)

[Hình 3.1 Sơ đồ khối tổng của hệ thống 5](file:///D:\Project\IR_Decode\document\bao_cao1.docx#_Toc502185046)

[Hình 3.2 Thiết kế khối nguồn 6](file:///D:\Project\IR_Decode\document\bao_cao1.docx#_Toc502185047)

[Hình 3.3 Thiết kế khối vi điều khiển 6](file:///D:\Project\IR_Decode\document\bao_cao1.docx#_Toc502185048)

[Hình 3.4 Thiết kế khối thu hồng ngoại 7](file:///D:\Project\IR_Decode\document\bao_cao1.docx#_Toc502185049)

[Hình 4.1 Lưu đồ giải thuật cho chương trình chính 9](#_Toc502185050)

[Hình 4.2 Lưu đồ giải thuật cho ngắt Tiemr Input capture 10](#_Toc502185051)

[Hình 4.3 Lưu đồ giải thuật cho ngắt Timer Update 11](#_Toc502185052)

[Hình 4.4 Lưu đồ giải thuật cho hàm xử lý sự kiện khi nhận được xung IR 12](#_Toc502185053)

[Hình 4.5 Lưu đồ giải thuật cho hàm giải mã tín hiệu IR remote Sony 13](#_Toc502185054)

[Hình 4.6 Lưu đồ giả thuật cho máy trạng thái IR command 14](#_Toc502185055)

[Hình 4.7 Luu đồ giải thuật cho máy trạng thái IR learning 15](#_Toc502185056)

[Hình 4.8 Lưu đồ giải thuật cho ngắt UART 16](#_Toc502185057)

[Hình 4.9 Lưu đồ giải thuật cho chương trình C# trên máy tính 16](#_Toc502185058)

[Hình 5.1 Phần cứng khi thực hiện xong 18](#_Toc502185059)

[Hình 5.2 Chức năng Debug của Keil 5 19](#_Toc502185060)

[Hình 5.3 Kết quả về độ rộng xung của xung Start khi nhấn nút Power 19](#_Toc502185061)

[Hình 5.4 Kết quả trả về của hàm IR\_Decode khi nhấn nút 1 19](#_Toc502185062)

[Hình 5.5 Xung tín hiệu IR quan sát được từ máy hiển thị song 20](#_Toc502185063)

[Hình 5.6 Đo độ rộng xung Start của tín hiệu khi nhấn nút 1 20](#_Toc502185064)

[Hình 5.7 Chương trình C# trên máy tính, hiển thị kết quả giải mã 21](#_Toc502185065)

[Hình 8.1 Thiết kế phần cứng 22](#_Toc502185066)

[Hình 8.2 Kết quả trả về của hàm IR\_Decode khi nhấn nút Power 30](#_Toc502185067)

[Hình 8.3 Kết quả trả về của hàm IR\_Decode khi nhấn nút Volume + 30](#_Toc502185068)

[Hình 8.4 Kết quả trả về của hàm IR\_Decode khi nhấn nút Channel + 30](#_Toc502185069)

[Hình 8.5 Kết quả trả về của hàm IR\_Decode khi nhấn nút Sound Mode 31](#_Toc502185070)

[Hình 8.6 Kết quả trả về của hàm IR\_Decode khi nhấn nút Mute 31](#_Toc502185071)

[Hình 8.7 Kết quả trả về của hàm IR\_Decode khi nhấn nút Space Sound 31](#_Toc502185072)

[Hình 8.8 Dạng xung của tín hiệu nút Volume + 32](file:///D:\Project\IR_Decode\document\bao_cao1.docx#_Toc502185073)

**DANH SÁCH BẢNG SỐ LIỆU**

[Bảng 1 Bảng mã lệnh của remote TV Sony RM 969 23](#_Toc502185033)

# GIỚI THIỆU

## Tổng quan

Tín hiệu hồng ngoại từ mỗi remote TV của mỗi hãng sản xuất có kiểu điều chế, mã hóa tín hiệu riêng.

Giải mã tín hiệu hồng ngoại từ remote TV bao gồm việc thu tín hiệu, giải mã các giao thức của từng chuẩn remote, xác định tín hiệu mang dữ liệu gì, từ đó hiển thị ra kết quả

## Nhiệm vụ đề tài

Nhiệm vụ của đề tài bao gồm tìm hiểu tín hiệu hồng ngoại của 1 số loại remote TV, dùng vi xử lý để giải mã từng loại và hiển thị kết quả. Các nhiệm vụ cụ thể như sau:

Nhiệm vụ 1: Tìm hiểu về nguyên lý, lý thuyết hoạt động của tia hồng ngoại trong remote TV.

Nhiệm vụ 2: Tìm hiểu về giao thức phát xung hồng ngoại trong remote TV Sony RM 969

Nhiệm vụ 3: Tìm hiểu về cảm biến thu hồng ngoại TSOP18XX, vi điều khiển STM32F030F4P6

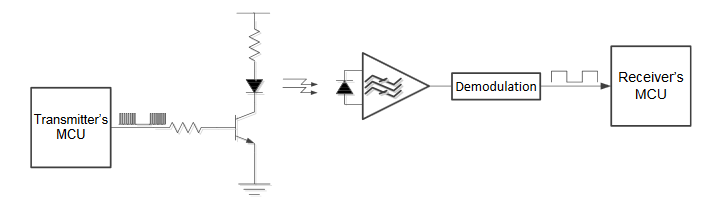
Nhiệm vụ 4: Thiết kế phần cứng dùng TSOP1838 để thu hồng ngoại và vi điều khiển STM32F030F4P6dể giải mã

Nhiệm vụ 5: Thiết kế hệ thống phần mềm để giải mã tín hiệu và hiển thị

# LÝ THUYẾT

## Tìm hiểu về tia hồng ngoại của remote TV

* Các remote hồng ngoại sử dụng ánh sáng hồng ngoại để truyền thông tin đến thiết bị.
* Ánh sáng hồng ngoại được điều chế và truyền đi bằng một IR LED trên thân remote. Do trong môi trường xung quanh có rất nhiều nguồn phát ra ánh sáng hồng ngoại, vì vậy việc điều chế là rất cần thiết, nó có tác dụng mã hóa dữ liệu và thay đổi tần số của ánh sáng hồng ngoại, giúp thiết bị thu có thể dễ dàng nhận biết được.
* Tần số điều chế thường dùng trong các remote TV thường từ 30kHz đến 60 kHz, phổ biến nhất là 38kHz[1]



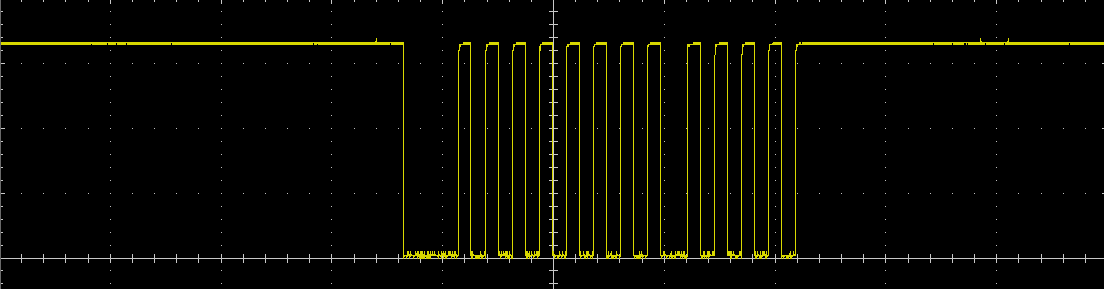
Hình 2.1 Ánh sáng hồng ngoại được điều chế và giải điều chế

## Tìm hiểu về LED thu hồng ngoại TSOP1838

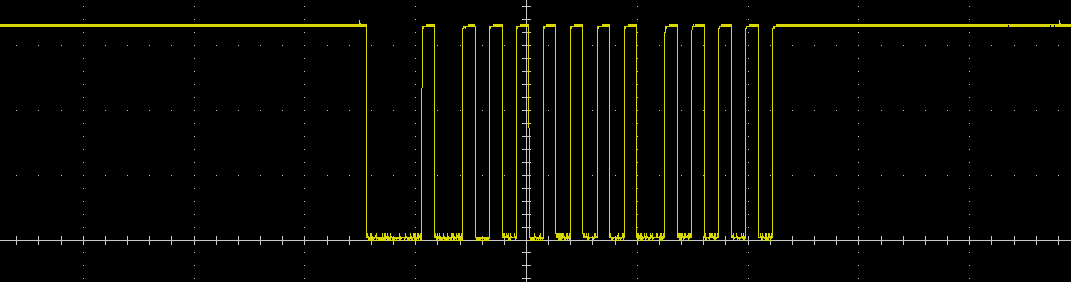
Hình . TSOP1838

* Tần số hoạt động: 38kHz
* Điện áp cung cấp: -0.3 đến 6V, phù hợp với mức điện áp 3.3V của vi điều khiển
* Ngõ ra tích cực thấp

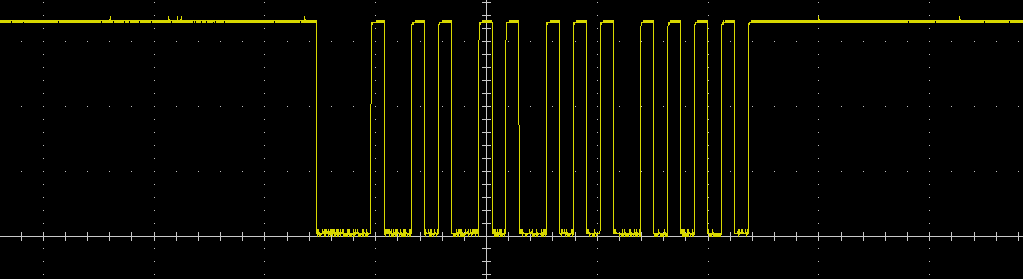
## Tìm hiểu về tín hiệu hồng ngoại từ remote TV Sony



Hình 2.3 Tín hiệu hồng ngoại thu được khi ấn nút “1” của remote TV Sony

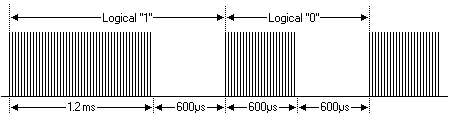


Hình 2.4 Tín hiệu hồng ngoại thu được khi ấn nút “2” của remote TV Sony

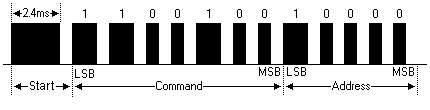


Hình 2.5 Tín hiệu hồng ngoại thu được khi ấn nút Power của remote TV Sony

* Hình 2.3, 2.4, 2.5 cho ta thấy được các xung tín hiệu trả về từ TSOP1838 khi có tín hiệu hồng ngoại từ remote TV Sony. Các mức tín hiệu mức cao ở 2 bên của hình là trạng thái trả khi không có tín hiệu. Các xung ở giữa hình chính là tín hiệu hồng ngoại đọc được
* Trong cả 3 hình, xung mức thấp thu dược đầu tiên luôn có độ rộng lớn nhất (khoảng 2400us) so với các xung mức thấp còn lại, đó là xung báo hiệu bắt đầu
* Các xung mức thấp có độ rộng nhỏ thứ 2 (khoảng 1200us) là xung báo tín hiệu 1, xung có độ rộng nhỏ nhất (khoảng 600us) là xung báo tín hiệu mức 0
* Giữa các xung mức thấp luôn có 1 xung mức cao có độ rộng bằng nhau (khoảng 600us)



Hình 2.6 Xung tín hiệu 1 và tín hiệu 0 của remote TV Sony. Nguồn: http://www.sbprojects.com/knowledge/ir/

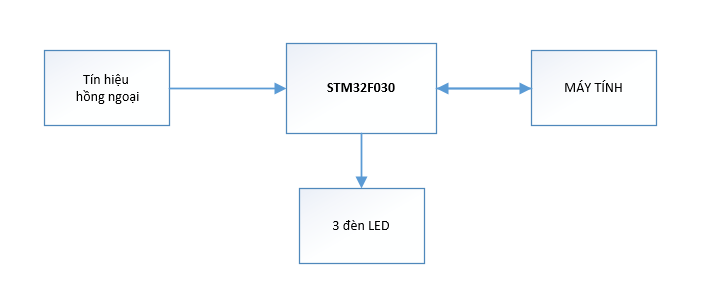


Hình 2.7 Tổng quát 1 chuỗi xung nhận được từ remote TV Sony. Nguồn: http://www.sbprojects.com/knowledge/ir/

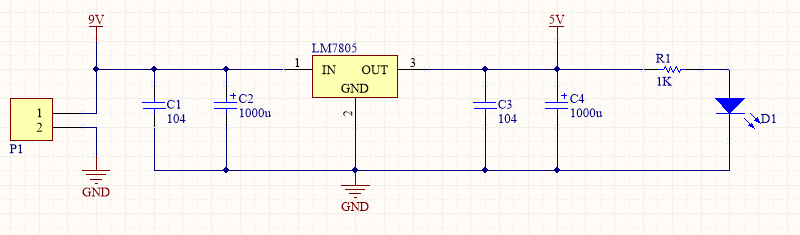
* 1 xung hồng ngoại từ remote TV mang 3 thành phần:
* Xung mở đầu: báo hiệu bắt đầu
* Chuỗi xung phân biệt các nút nhấn trên remote (command)
* Chuỗi xung phân biệt loại remote là remote TV hay remote máy lạnh,… (address)
* Khi ấn đè 1 nút thì chum xung sẽ được phát ra liên tục, khoảng cách từ cạnh xuống xung bắt đầu của chùm xung trước đến cạnh xuống của xung bắt đầu chùm xung sau là khoảng 44ms
* Mỗi loại remote, mỗi nhà sản xuất có một cách mã hóa tín hiệu riêng

# THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

* **Yêu cầu thiết kế**
  + Phần cứng thực hiện được nhiệm vụ đề tài là thu và giải mã tín hiệu hồng ngoại của remote TV
  + Phải có phần hiển thị kết quả giải mã được
  + Có thêm một phần để áp dụng việc giải mã tín hiệu vào điều khiển
* **Phân tích thiết kế**
* Sử dụng LED thu hồng ngoại TSOP1838 để nhận tín hiệu hồng ngoại
* Vi điều khiển: STM32F030F4P6 để thực hiện giải mã tín hiệu, điều khiển thiết bị và hiển thị kết quả
* Thiết bị chọn để điều khiển là 3 đèn LED để có thể quan sát trực quan
* Hiển thị kết quả: kết quả giải mã được sẽ được gửi lên máy tính thông qua module UART - COM.

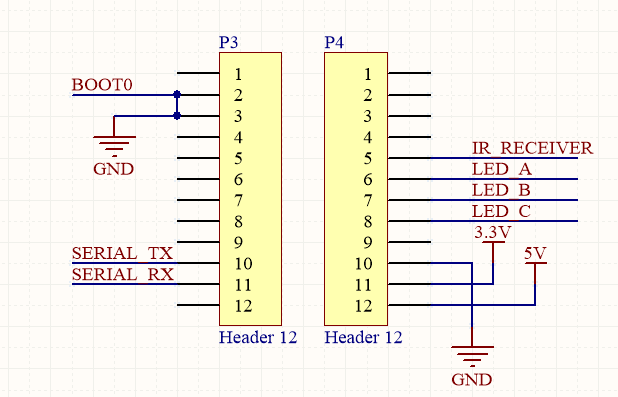


Hình . Sơ đồ khối tổng của hệ thống

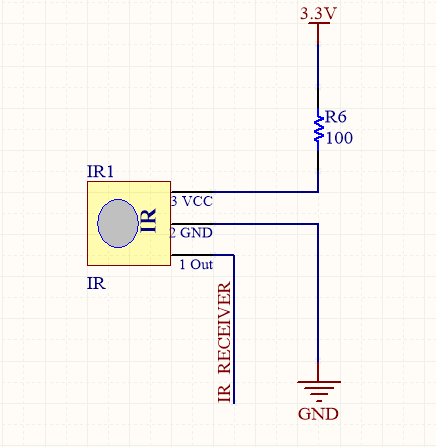


Hình . Thiết kế khối nguồn

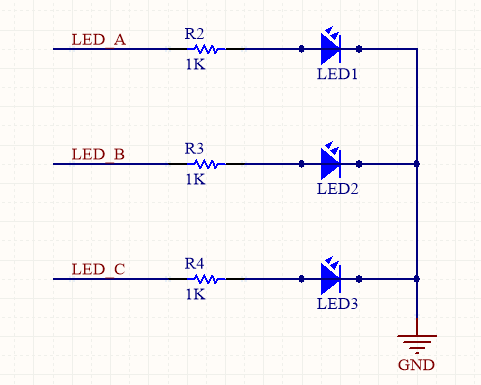
* Trong khối nguồn, ta sử dụng IC LM7805 để tạo điện áp 5V ổn định, các tụ điện C1, C2, C3, C4 để lọc nhiễu. LED D1 để báo nguồn
* Chân 5V và GND để cấp nguồn cho STM32F030



Hình . Thiết kế khối vi điều khiển

* Trong khối vi điều khiển:
* Chân BOOT0 cần nối xuống GND để không đưa STM32F030 và trạng thái Bootloader
* Các chân SERIAL\_TX, SERIAL\_RX để kết nối đến máy tính qua module UART – COM.
* IR\_RECEIVER nối đến chân OUT của TSOP1838 để nhận tín hiệu hồng ngoại.
* LED\_A, LED\_B, LED\_C là 3 chân GPIO điều khiển LED.
* Chân 3.3V để cấp nguồn 3.3V cho TSOP1838.

Hình . Thiết kế khối thu hồng ngoại

* Trong khối thu hồng ngoại, chân Out của TSOP1838 sẽ kết nối đến GPIO trên STM32F030. Chân VCC sẽ nối đến nguồn 3.3V. điện trở R6 có tác dụng hạn dòng, bảo vệ LED thu hồng ngoại

Hình .5 Thiết kế khối đèn LED

# THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

* Yêu cầu đặt ra cho phần mềm
  + Phần mềm cho STM32F030:
* Nhận tín hiệu hồng ngoại:
  + - Sử dụng chức năng Input capture tại PB1 (Timer 3 channel 4) để tính thời gian giữa các xung từ LED thu hồng ngoại (độ chính xác là micro giây).
    - Việc bắt các xung sẽ sử dụng ngắt để đảm bảo độ chính xác
* Gửi và nhận dữ liệu đến máy tính, sử dụng UART1 với thông số:
  + - Tx Pin: PA2
    - Rx Pin: PA3
    - Baudrate: 115200
    - Stop bit: 1
    - Parity: không
    - Sử dụng ngắt để nhận dữ liệu từ UART
* 3 đèn LED dùng để điều khiển nối vào 3 GPIO PA5, PA6, PA7



Hình 4.1 Lưu đồ giải thuật cho chương trình chính



Hình 4.2 Lưu đồ giải thuật cho ngắt Tiemr Input capture



Hình 4.3 Lưu đồ giải thuật cho ngắt Timer Update



Hình 4.4 Lưu đồ giải thuật cho hàm xử lý sự kiện khi nhận được xung IR



Hình 4.5 Lưu đồ giải thuật cho hàm giải mã tín hiệu IR remote Sony



Hình 4.6 Lưu đồ giả thuật cho máy trạng thái IR command



Hình 4.7 Luu đồ giải thuật cho máy trạng thái IR learning



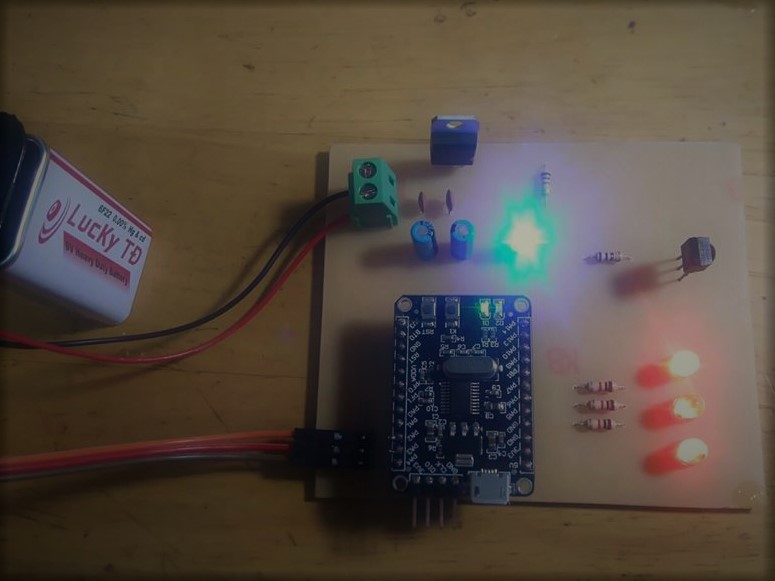
Hình 4.8 Lưu đồ giải thuật cho ngắt UART



Hình 4.9 Lưu đồ giải thuật cho chương trình C# trên máy tính

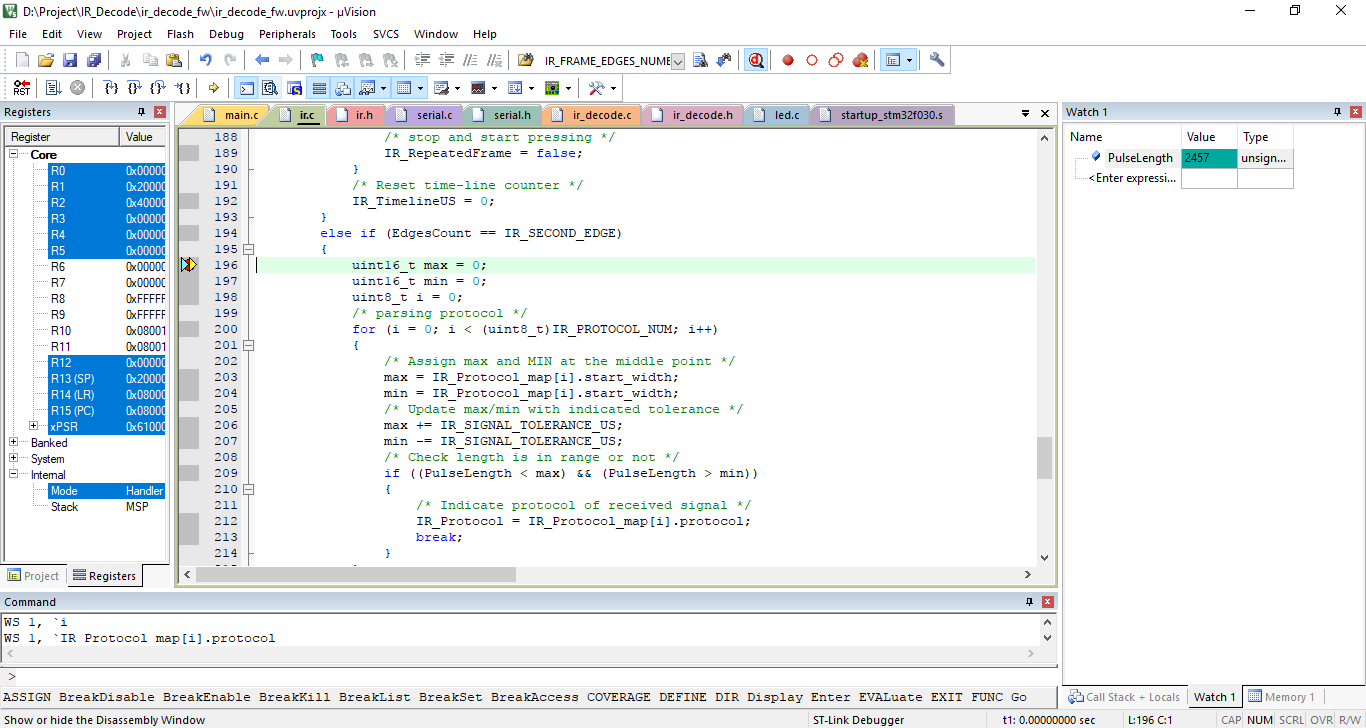
* Hệ thống cho vi điều khiển sử dụng máy trạng thái với 2 trạng thái chính:
* IR command state: STM32F030 sẽ nhận lệnh từ remote TV Sony, gửi mã lệnh qua UART lên máy tính và kiểm tra với các lệnh điều khiển 3 LED, nếu đúng thì sẽ đổi trạng thái của LED. Lệnh điều khiển 3 LED được lưu trong 3 biến tĩnh.
* IR Learning state: khi nhận được lệnh gửi từ UART (0x65), chương trình sẽ đổi sang trạng thái học lệnh, STM32F030 sẽ đợi nhận 3 lệnh từ remote TV và lưu 3 lệnh này vào 3 biến lưu mã lệnh điều khiển đèn LED. Sau khi hoàn thành, trạng thái sẽ được đổi qua IR command state.
* Việc nhận và giải mã tín hiệu hồng ngoại sẽ được thực hiện bằng ngắt. Các giá trị độ rộng xung, số thứ tự cạnh và giao thức hồng ngoại của remote sẽ được đưa vào hàm IR\_Decode để giải mã dữ liệu.
* Trong hàm IR\_Deocde, nếu giải mã thành công tín hiệu IR, hàm sẽ trả về giá trị dữ liệu IR, mã lệnh và địa chỉ giải mã được.

# KẾT QUẢ THỰC HIỆN

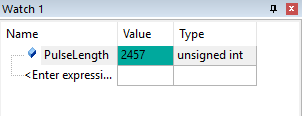


Hình 5.1 Phần cứng khi thực hiện xong

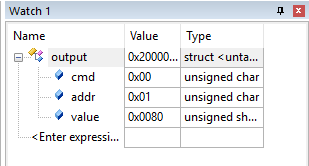
* Để kiểm tra nguồn cho phần cứng, dùng VOM để đo điện áp tại chân Out của IC LM7805, nếu điện áp 5V thì đã khối nguồn đã hoạt động đúng.
* Để kiểm tra xung tín hiệu hồng ngoại từ TSOP1838, dùng máy hiển thị sóng Hantek 6022be để hiển thị và kiểm chứng
* Để kiểm tra kết quả của việc giải mã tín hiệu, em sử dụng bộ Debug của Keil 5



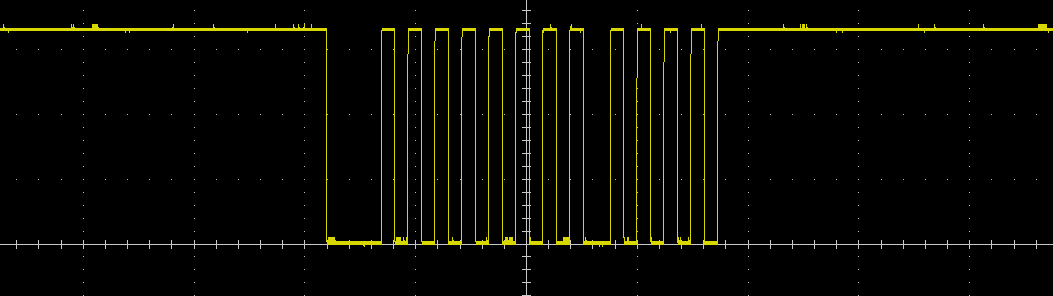
Hình . Chức năng Debug của Keil 5



Hình . Kết quả về độ rộng xung của xung Start khi nhấn nút Power

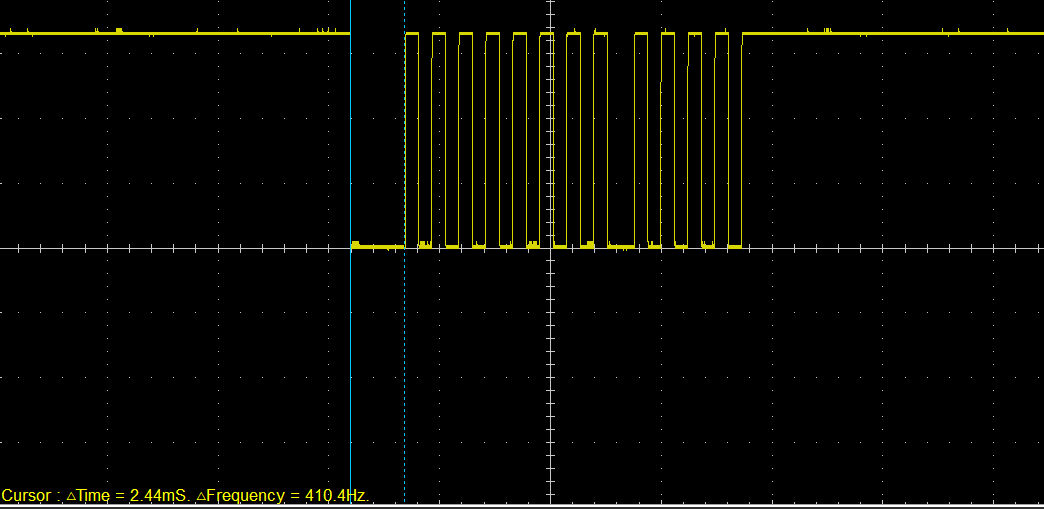


Hình . Kết quả trả về của hàm IR\_Decode khi nhấn nút 1



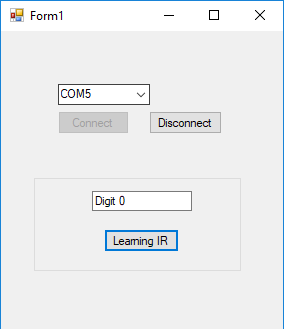
Hình . Xung tín hiệu IR quan sát được từ máy hiển thị song

Dùng chức năng Cursor trên phần mềm của máy hiển thị song để đo độ rộng xung



Hình . Đo độ rộng xung Start của tín hiệu khi nhấn nút 1

* Thực hiện kết nối với máy tính qua module USB-UART CP2102



Hình . Chương trình C# trên máy tính, hiển thị kết quả giải mã

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

* Đạt được yêu cầu đặt ra: giải mã được tín hiệu hồng ngoại từ remote TV Sony RM 969
* Hạn chế: việc giải mã tín hiệu đôi khi không được nhạy, còn hạn chế trong số lượng remote có thể giải mã

Trong quá trình làm đồ án, gặp nhiều vấn đề thực tiển khác với lý thuyết đã học buộc em phải tìm cách khắc phục nó, từ đó bổ sung nhiều phần hổng trong quá trình học tập. Hiểu rõ quá trình thiết kế, lập trình một hệ thống đơn giản.

## Hướng phát triển

Dựa vào cách giải mã tín hiệu hồng ngoại remote TV SonyRM 969, có thể phát triển để giải mã các loại remote khác, đạt được mục tiêu có thể nhận được tín hiệu từ nhiều loại remote, ứng dụng vào điều khiển thông minh.

# TÀI LIỆU THAM KHẢO

[1] Texas Instrument, *Infrared Remote Control Implementation With MSP430FR4xx* (SLAA644B*)*

[2] Vishay, Photo Modules for PCM Remote Control Systems, TSOP18xx datasheet, Sep. 2000

[3] STMicroelectronics, STM32F030x4/x6/x8/xC and STM32F070x6/xB

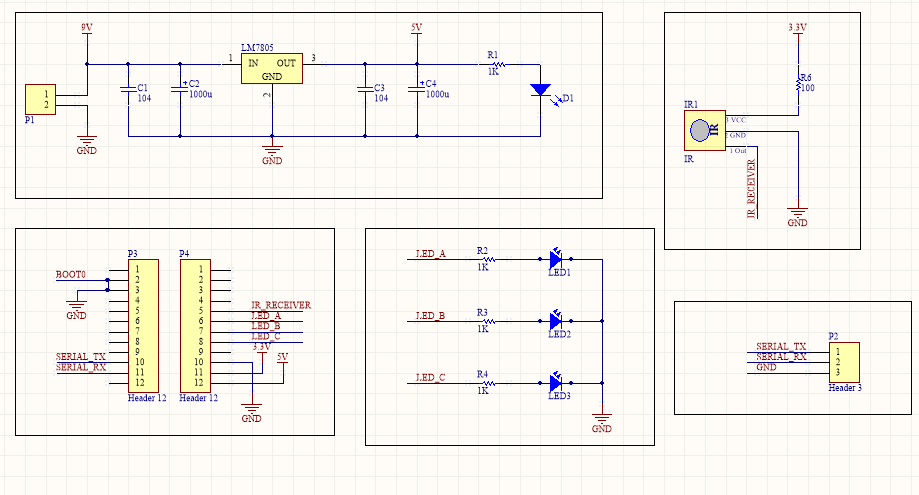
advanced ARM®-based 32-bit MCUs, Reference Manual, April. 2017

[4] STMicroelectronics, STM32F030x4 datasheet, Jan. 2017

[5] Hoàng Trang, Bùi Quốc Bảo, *Lập trình hệ thống nhúng,* NXB Đại Học Quốc Gia TP.HCM, 2016

[6] Tài liệu tham kháo khác: http://www.sbprojects.com/knowledge/ir/

# PHỤ LỤC



Hình 8.1 Thiết kế phần cứng

|  |  |
| --- | --- |
| Mã lệnh (hex) | Nút nhấn |
| 0x14 | Mute |
| 0x3A | i+ |
| 0x15 | Power |
| 0x3F | Menu |
| 0x25 | Video mode |
| 0x38 | Menu off |
| 0x00 | 1 |
| 0x01 | 2 |
| 0x02 | 3 |
| 0x03 | 4 |
| 0x04 | 5 |
| 0x05 | 6 |
| 0x06 | 7 |
| 0x07 | 8 |
| 0x08 | 9 |
| 0x09 | 0 |
| 0x3B | Jump |
| 0x1D | -/-- |
| 0x28 | Sound Mode |
| 0x12 | Volume + |
| 0x13 | Volume - |
| 0x10 | Channel + |
| 0x11 | Channel - |
| 0x17 | A/B |
| 0x3C | Timer On |
| 0x36 | Timer Off |
| 0x74 | Select Left |
| 0x7C | Select Right |
| 0x75 | PIC Mode Left |
| 0x64 | PIC Mode Right |
| 0x29 | Space Sound |

Bảng 1 Bảng mã lệnh của remote TV Sony RM 969

* Hàm giải mã tín hiệu IR:

int IR\_Decode (ir\_decode\_intput\_t \*input, ir\_decode\_output\_t \*output)

{

static ir\_decode\_var\_t var;

switch (input->protocol)

{

case IR\_PROTOCOL\_SONY\_SIRC:

return IR\_SonyDecode(\*input, output, &var);

default:

break;

}

return 1;

}

* Hàm giải mã tín hiệu cho remote TV Sony:

uint32\_t IR\_SonyDecode(ir\_decode\_intput\_t input, ir\_decode\_output\_t\* output, ir\_decode\_var\_t\* var)

{

/\* SONY transmit LSB first \*/

uint16\_t vLongMax, vLongMin;

vLongMax = IR\_SONY\_PROTOCOL\_LONG\_INTERVAL + IR\_SIGNAL\_TOLERANCE\_US;

vLongMin = IR\_SONY\_PROTOCOL\_LONG\_INTERVAL - IR\_SIGNAL\_TOLERANCE\_US;

if (input.edges == 3){

var->xCount = 0;

var->xPreviousBit = 0;

output->value = 0;

}

else

{

if ((input.edges & 0x01) == 0) //!< Data is available at even edge

{

/\* Increase bits count \*/

var->xCount++;

/\* Shift out to convert from LSB to MSB and the end of frame \*/

var->xPreviousBit = var->xPreviousBit >> 1;

/\* Check 1 bit \*/

if ((input.interval > vLongMin) && (input.interval < vLongMax)){

var->xPreviousBit += ((uint16\_t)1<<11);

}

/\* Check whether output is available or not \*/

if (var->xCount == 12) {

output->value = var->xPreviousBit;

/\* parse IR value to command and address \*/

output->cmd = output->value & 0x7F;

output->addr = (output->value >> 7) & 0x1F;

return 0;

}

}

}

return 1;

}

* Hàm ngắt Timer:

void TIM3\_IRQHandler (void)

{

static uint8\_t EdgesCount = 0;

static uint32\_t IR\_TimelineUS = 0;

static bool IR\_RepeatedFrame = false;

static int temp = 0;

uint32\_t PulseLength = 0;

ir\_event\_t Event = IR\_EVENT\_ERROR\_RECEIVED\_UNKNOWN\_PROTOCOL;

/\* time overflow event \*/

if (SET == TIM\_GetITStatus(TIM3, TIM\_IT\_Update))

{

/\* update timeline \*/

IR\_TimelineUS += 65535;

if (IR\_TimelineUS >= 500000)

{

EdgesCount = 0;

IR\_TimelineUS = 0;

}

/\* Clear interrupt flag of timer overflow \*/

TIM\_ClearITPendingBit(TIM3, TIM\_IT\_Update);

}

/\* Input capture event \*/

if (SET == TIM\_GetITStatus(TIM3, TIM\_IT\_CC4))

{

/\* Increase edges counter \*/

EdgesCount++;

/\* Get timer counter \*/

PulseLength = (uint32\_t)(TIM\_GetCapture4(TIM3));

/\* indicate event \*/

Event = IR\_EVENT\_INPUT\_SIGNAL;

/\* increase timeline \*/

IR\_TimelineUS += PulseLength;

/\* check EdgesCount \*/

if (EdgesCount == IR\_FIRST\_EDGE)

{

/\* First edge. indicate start of a frame \*/

/\* Checking repeated frame condition \*/

if (IR\_TimelineUS < IR\_SONY\_START\_TO\_START\_INTERVAL\_US)

{

/\* Repeated frame \*/

IR\_RepeatedFrame = true;

} else

{

/\* stop and start pressing \*/

IR\_RepeatedFrame = false;

}

/\* Reset time-line counter \*/

IR\_TimelineUS = 0;

}

else if (EdgesCount == IR\_SECOND\_EDGE)

{

uint16\_t max = 0;

uint16\_t min = 0;

uint8\_t i = 0;

/\* parsing protocol \*/

for (i = 0; i < (uint8\_t)IR\_PROTOCOL\_NUM; i++)

{

/\* Assign max and MIN at the middle point \*/

max = IR\_Protocol\_map[i].start\_width;

min = IR\_Protocol\_map[i].start\_width;

/\* Update max/min with indicated tolerance \*/

max += IR\_SIGNAL\_TOLERANCE\_US;

min -= IR\_SIGNAL\_TOLERANCE\_US;

/\* Check length is in range or not \*/

if ((PulseLength < max) && (PulseLength > min))

{

/\* Indicate protocol of received signal \*/

IR\_Protocol = IR\_Protocol\_map[i].protocol;

break;

}

}

if ((i == (uint8\_t)IR\_PROTOCOL\_NUM) && ir\_callback){

/\* Indicate unknown protocol \*/

IR\_Protocol = IR\_PROTOCOL\_NUM;

/\* Indicate error event \*/

Event = IR\_EVENT\_ERROR\_RECEIVED\_UNKNOWN\_PROTOCOL;

}

}

else

{

/\* not first or second edge \*/

if (IR\_RepeatedFrame == false)

{

if (ir\_callback != ((void\*)0))

/\* call callback handler \*/

ir\_callback(Event, PulseLength, EdgesCount, IR\_Protocol);

}

if ((IR\_Protocol == IR\_PROTOCOL\_SONY\_SIRC)

&& (EdgesCount == IR\_FRAME\_EDGES\_NUMER\_SONY\_SIRC\_PROTOCOL)){

/\* Self reset edges counter \*/

EdgesCount = 0;

}

}

/\* Clear interrupt flag \*/

TIM\_ClearFlag(TIM3, TIM\_FLAG\_CC4);

/\* Set the counter value to zero \*/

TIM\_SetCounter(TIM3, 0);

}

}

* Hàm IR\_EventHandle

static void IR\_EventHandle (ir\_event\_t xEvent, uint16\_t xInterval, \

uint8\_t xEdgesCount, ir\_protocol\_t xIrProtocol)

{

ir\_decode\_intput\_t pcIn;

pcIn.edges = xEdgesCount;

pcIn.interval = xInterval;

pcIn.protocol = xIrProtocol;

/\* call decode function \*/

if (IR\_Decode(&pcIn, &IR\_Output) == 0)

{

isGetNewSignal = true;

}

}

* Chương trình chính:

int main (void)

{

uint8\_t cmd;

SetupHardware();

while (1)

{

switch (gState)

{

case STATE\_CMD:

if (IR\_GetCmd(&cmd) == true)

{

IR\_ClearRecvFlag();

Serial\_SendByte(cmd);

if (cmd == IR\_cmd\_LED\_A)

Led\_Toggle(LED\_A);

if (cmd == IR\_cmd\_LED\_B)

Led\_Toggle(LED\_B);

if (cmd == IR\_cmd\_LED\_C)

Led\_Toggle(LED\_C);

}

break;

case STATE\_LEARNING:

Led\_SetLevel(LED\_A, LED\_LEVEL\_ENABLE);

Led\_SetLevel(LED\_B, LED\_LEVEL\_ENABLE);

Led\_SetLevel(LED\_C, LED\_LEVEL\_ENABLE);

while(IR\_GetCmd(&cmd) == false);

IR\_ClearRecvFlag();

IR\_cmd\_LED\_A = cmd;

Led\_SetLevel(LED\_A, LED\_LEVEL\_DISABLE);

while(IR\_GetCmd(&cmd) == false);

IR\_ClearRecvFlag();

IR\_cmd\_LED\_B = cmd;

Led\_SetLevel(LED\_B, LED\_LEVEL\_DISABLE);

while(IR\_GetCmd(&cmd) == false);

IR\_ClearRecvFlag();

IR\_cmd\_LED\_C = cmd;

Led\_SetLevel(LED\_C, LED\_LEVEL\_DISABLE);

gState = STATE\_CMD;

/\* Send 'f' ACK to app: done learing IR \*/

Serial\_SendByte('f');

break;

default:

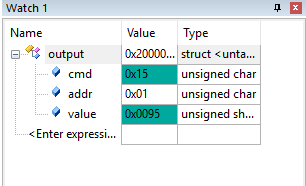
break;

}

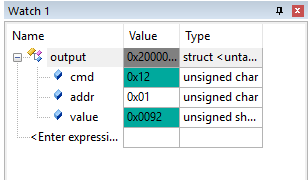
}

}

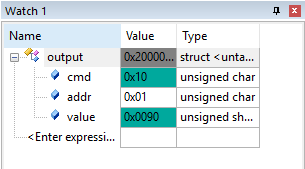
* Hình ảnh kết quả trả về trên Debug Keil 5 của hàm IR\_Decode



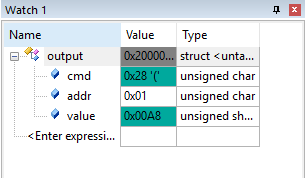
Hình . Kết quả trả về của hàm IR\_Decode khi nhấn nút Power



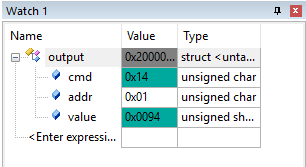
Hình . Kết quả trả về của hàm IR\_Decode khi nhấn nút Volume +



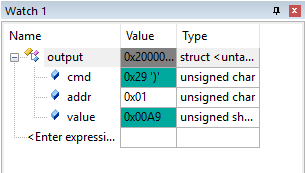
Hình . Kết quả trả về của hàm IR\_Decode khi nhấn nút Channel +



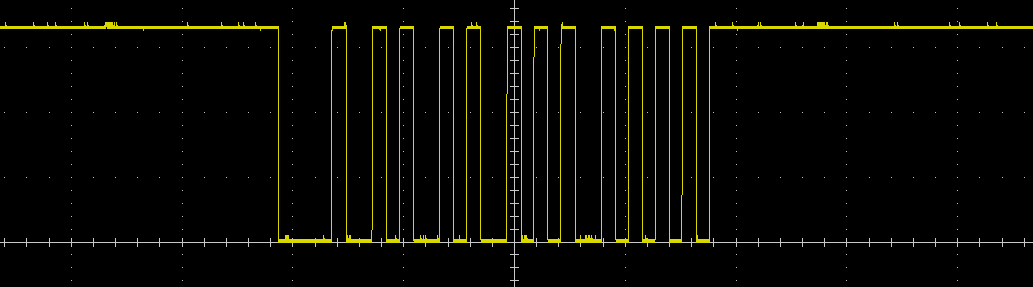
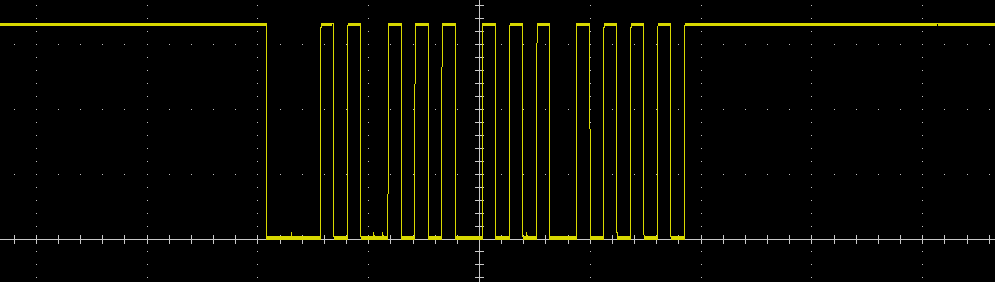
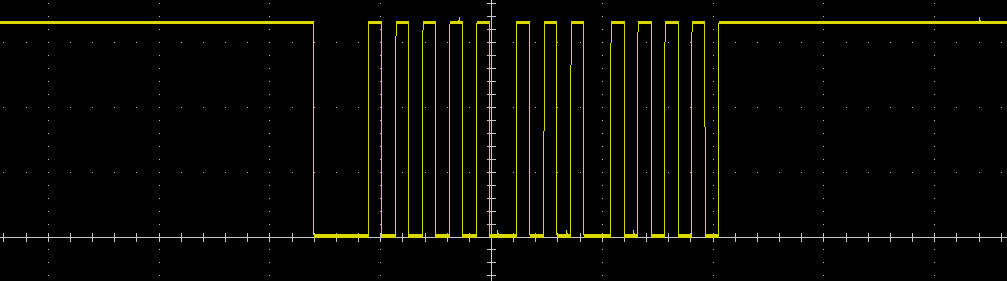
Hình . Kết quả trả về của hàm IR\_Decode khi nhấn nút Sound Mode



Hình . Kết quả trả về của hàm IR\_Decode khi nhấn nút Mute



Hình . Kết quả trả về của hàm IR\_Decode khi nhấn nút Space Sound

* Hình ảnh dạng xung từ máy hiển thị sóng

Hình .10 Dạng xung của tín hiệu nút Channel +

Hình . Dạng xung của tín hiệu nút Volume +

Hình .8 Dạng xung của tín hiệu nút Power