1. 과 제 명 : 컬러센서를 활용한 자동 물류 분류 시스템

2. 시스템 구현 목표

- 컬러 센서를 이용한 모터 제어 기능 구현

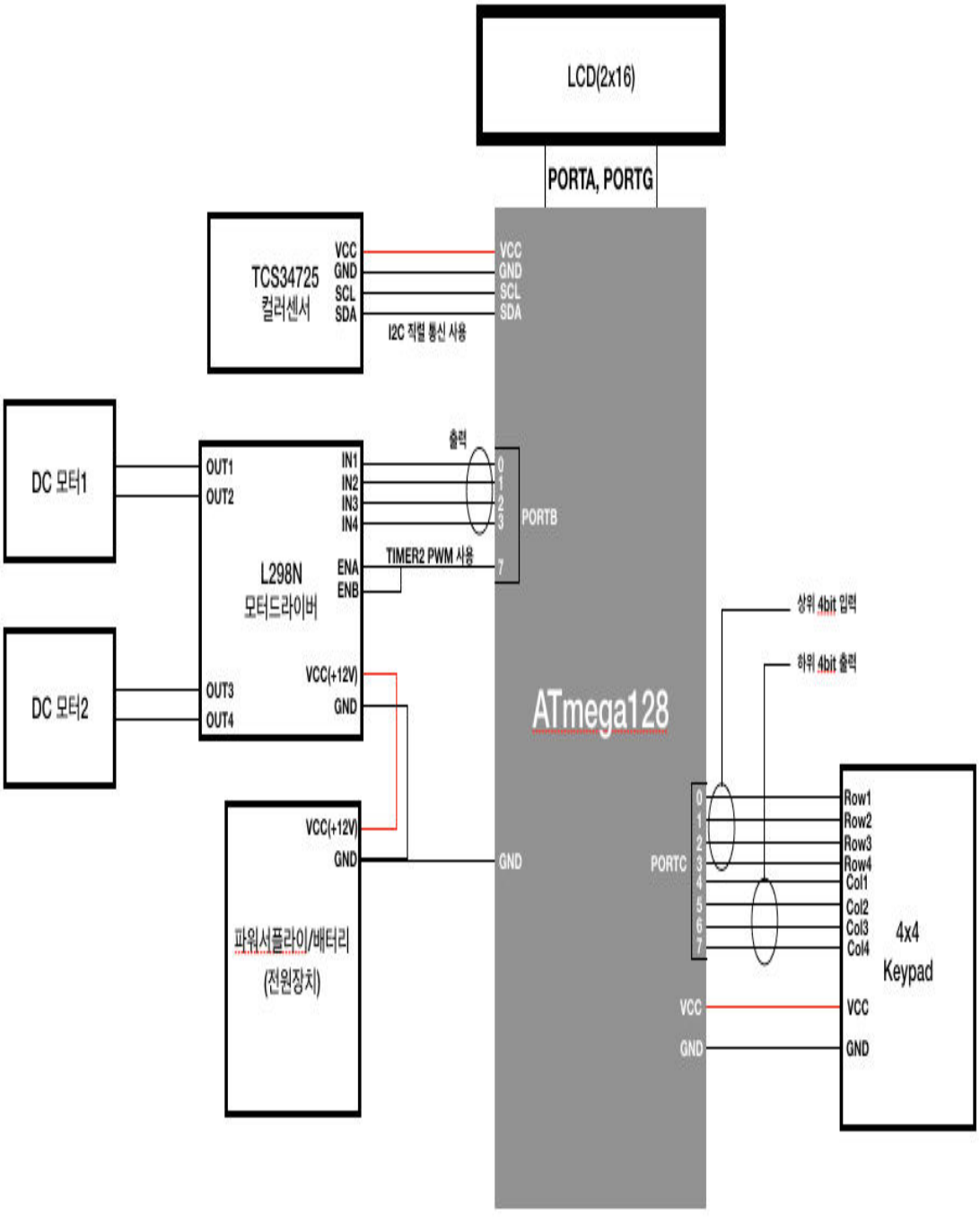
  컬러센서에서 RGBC 값을 받아와 상자 색깔과 유무에 따라 모터를 제어한다.

- 모터 드라이버를 이용한 모터 제어 기능 구현
- 키패드를 이용한 모터 제어 기능 구현

  키패들를 이용하여 시스템의 모드 및 모터를 제어한다.

3. H/W 시스템 구성

3.1 H/W 시스템 블록다이어그램



3.2 각각의 H/W의 역할 및 인터페이스 방법

- **ATmega128** : 본 시스템을 총괄하는 역할을 하는 MCU이다. 필요한 센서들간의 통신을 실시하며 조건에 맞는 출력을 내보냄으로서 시스템이 작동할 수 있도록 하는 매우 중요한 역할을 담당한다.

- **LCD** : 사용자가 본 시스템이 어떠한 일을 하는지 표시해주는 표시장치로서 역할을 한다. 2x16 으로 구성되어 있다.

- **4x4 키패드** : 본 시스템에서 사용자가 직접 조작할 수 있는 부분이다. 키배열은 4x4로 16개의 버튼을 누를 수 있다. 본 시스템에서는 모드를 선택 받을 수 있으며 수동모드시 키패드로 모터를 제어 할 수 있다.

- **TCS34725(컬러센서)** : 본 시스템의 자동모드일 때 입력을 담당한다. MCU와의 통신은 I2C 직렬 통신을 사용한다. 색깔에 따라 RGB값을 반환하며 clear값도 반환하여 물체가 멀거나 없을 때를 구분이 가능하다.

- **L298N(모터드라이버)** : 2개의 DC 모터를 제어하는 역할을 담당한다. MCU로부터 PWM 신호와 모터의 출력과 관련된 신호를 받아 모터를 제어한다.
  전원은 파워서플라이나 배터리 등에서 받아와 모터를 제어한다.

- **파워서플라이/배터리(전원장치)** : 모터를 구동하기 위하여 전원을 공급하는 역할을 담당한다.

3.3 사용 센서에 대한 사양 분석 및 고려 사항

본 프로젝트를 진행함에 있어서 가장 시간이 많이 투자 된 곳이 TCS34725를 분석하고 사용하는 것 이었다. 모터드라이버와 키패드는 비교적 쉽게 분석하고 사용 할 수 있었다.

– TCS34725(컬러센서)

센서개요 : TCS34725에는 3 × 4 광다이오드, 광다이오드 전류, 데이터 레지스터, 상태 레지스터를 통합하는 4개의 아날로그–디지털 변환기(ADC)가 있다. 3×4 광다이오드는 빨간색, 녹색 필터링, 파란색, 투명 광다이오드로 구성된다. 통합형 ADC 4개는 동시에 증폭된 광다이오드 전류를 16비트 디지털 값으로 변환한다. TCS34725의 데이터 통신은 최대 400kHz의 I2C 직렬 버스를통해 이루어진다. 별도의 인터럽트 신호 출력이 가능하여 폴링 방식을 사용하지 않고 사용할 수 있다.

레지스터 설정 : 직렬 인터페이스를 통해 레지스터에 접속하여 제어한다. 다음과 같은 레지스터들을 설정할 수 있다.

Table 3. Register Address

ADDRESS	RESISTER NAME	R/W	REGISTER FUNCTION	RESET VALUE
--	COMMAND	W	Specifies register address	0x00
0x00	ENABLE	R/W	Enables states and interrupts	0x00
0x01	ATIME	R/W	RGBC time	0xFF
0x03	WTIME	R/W	Wait time	0xFF
0x04	AILTL	R/W	Clear interrupt low threshold low byte	0x00
0x05	AILTH	R/W	Clear interrupt low threshold high byte	0x00
0x06	AIHTL	R/W	Clear interrupt high threshold low byte	0x00
0x07	AIHTH	R/W	Clear interrupt high threshold high byte	0x00
0x0C	PERS	R/W	Interrupt persistence filter	0x00
0x0D	CONFIG	R/W	Configuration	0x00
0x0F	CONTROL	R/W	Control	0x00
0x12	ID	R	Device ID	ID
0x13	STATUS	R	Device status	0x00
0x14	CDATAL	R	Clear data low byte	0x00
0x15	CDATAH	R	Clear data high byte	0x00
0x16	RDATAL	R	Red data low byte	0x00
0x17	RDATAH	R	Red data high byte	0x00
0x18	GDATAL	R	Green data low byte	0x00
0x19	GDATAH	R	Green data high byte	0x00
0x1A	BDATAL	R	Blue data low byte	0x00
0x1B	BDATAH	R	Blue data high byte	0x00

본 프로젝트에서 주로 사용할 레지스터는 Command 레지스터와 Enable 레지스터, RGBC 채널 데이터 레지스터 이다.

Commend 레지스터는 쓰기 및 읽기 작업에 사용할 대상 레지스터의 주소를 지정한다. 데이터 레지스터를 읽어오거나 상태, Enable 레지스터에 접속할 때 반드시 거쳐야 하는 레지스터이다.

Table 4. Command Register

	7	6	5	4	3	2	1	0	
COMMAND	CMD	TYPE		ADDR/SF					--

cmd 비트가 1이어야 사용이 가능하고 0~4번 비트에 주소를 지정한다.

Enable 레지스터는 TCS3425 장치의 전원을 켜고 끄는데 사용되며 다음 사진과 같은 기능과 인터럽트 활성화가 가능하다.

	7	6	5	4	3	2	1	0	
ENABLE	Reserved			AIEN	WEN	Reserved	AEN	PON	Address 0x00

FIELD	BITS	DESCRIPTION
Reserved	7:5	Reserved. Write as 0.
AIEN	4	RGBC interrupt enable. When asserted, permits RGBC interrupts to be generated.
WEN	3	Wait enable. This bit activates the wait feature. Writing a 1 activates the wait timer. Writing a 0 disables the wait timer.
Reserved	2	Reserved. Write as 0.
AEN	1	RGBC enable. This bit actives the two-channel ADC. Writing a 1 activates the RGBC. Writing a 0 disables the RGBC.
PON [1, 2]	0	Power ON. This bit activates the internal oscillator to permit the timers and ADC channels to operate. Writing a 1 activates the oscillator. Writing a 0 disables the oscillator.

본 프로젝트에서는 인터럽트 방식을 사용하지 않았으므로 PON, AEN비트만 활성화 시켜주었다.

RGBC 채널 데이터 레지스터는 Clear, Red, Green, Blue의 데이터가 16비트 값으로 저장된다. 그러므로 데이터가 올바르게 읽히려면 2바이트씩을 읽어와야 제대로된 값을 받아올 수 있다. 레지스터의 모습은 다음과 같다.

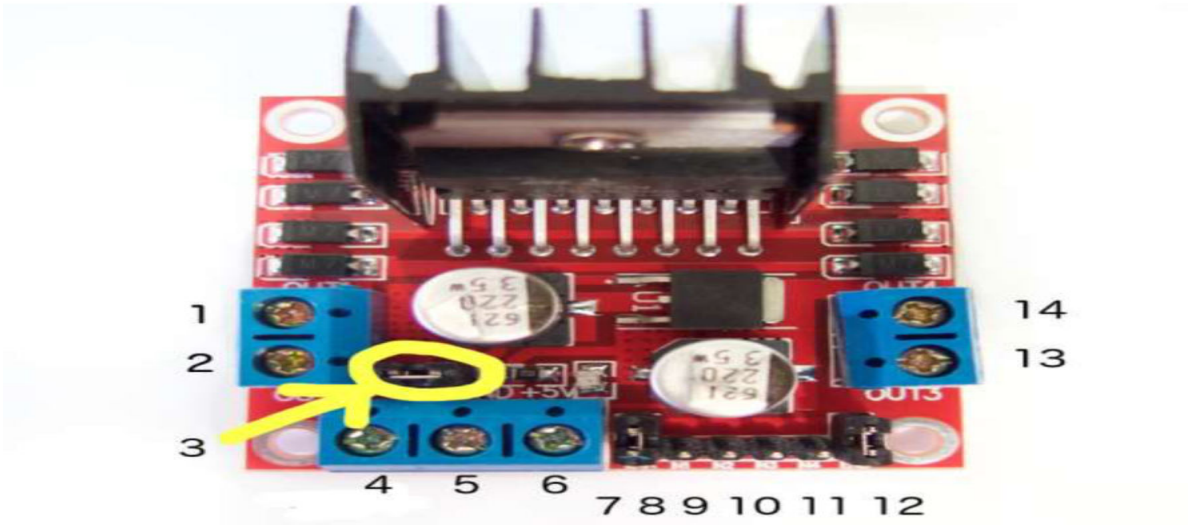
Table 14. ADC Channel Data Registers

REGISTER	ADDRESS	BITS	DESCRIPTION
CDATA	0x14	7:0	Clear data low byte
CDATAH	0x15	7:0	Clear data high byte
RDATA	0x16	7:0	Red data low byte
RDATAH	0x17	7:0	Red data high byte
GDATA	0x18	7:0	Green data low byte
GDATAH	0x19	7:0	Green data high byte
BDATA	0x1A	7:0	Blue data low byte
BDATAH	0x1B	7:0	Blue data high byte

TCS34725 센서를 사용할 때 고려사항으로서는 가장 RGBC 데이터를 2바이트씩 읽어와 처리하는 것에 있다. 이와 관련된 사항을 알지 못했다면 잘못된 데이터 값으로 처리해야 할 뻔했다. TCS34725는 빛을 사용하여서 데이터 값을 받아오므로 극심하게 차이나는 환경에서의 컬러 측정은 쉽지 않았다. 하지만 제한된 환경을 만들어주기만 하면 데이터 값은 상당히 정확하게 나오는 것을 확인 할 수 있었다.

– L298N(모터드라이버)

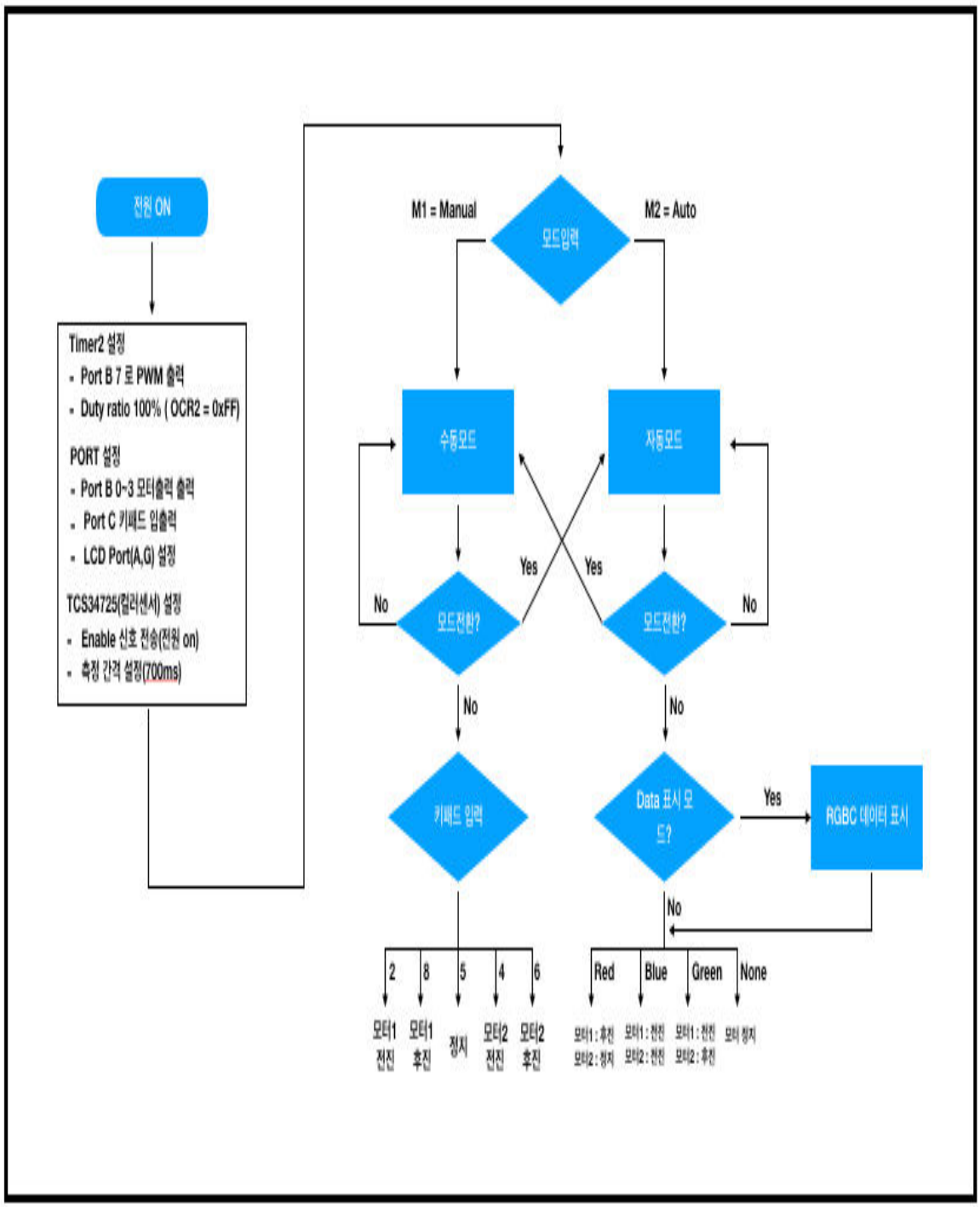
위 모듈을 사용할때에는 큰 어려움이 없었다. 다음의 그림대로 선을 입력하고 이곳에 비트를 설정해 주거나 PWM 신호를 입력해주면 되었다.



위 그림에서 다소 살펴봐야 할 부분은 7번과 12번 포트이다. 이곳은 PWM 신호를 넣어주는 곳이다.본 프로젝트에서는 두 모터 모두 Duty Ratio를 100%로 설정해 주었다. 그러한 이유는 컨베이어 벨트 제작시 완전히 1:1로 맞물리는 궤도를 사용한 것이 아니라 간이식으로 제작이 되었기 때문에 보다 원활한 작동을 위해서였다. 이 부분외에는 모터드라이브를 사용하는데에 큰 어려움은 없었다.

4. S/W 및 시나리오

4.1 시스템 Flow chart



4.2 시스템 작동에 관하여

1. 기본적인 초기 설정을 마친 후 모드 선택을 실시한다. 시스템의 모드는 수동과 자동 모드로 나누어 진다. 자동모드에서는 컬러센서의 데이터 값을 볼수 있는 옵션도 제공하고 있다.
2. 자동모드 선택시 컬러센서의 값으로 모터를 제어한다. 상자가 없다고 판단되면 작동을 하지 않는다.
3. 빨간색 상자가 감지될시 불량으로 인지하여 상자를 뒤로 보낸다.
4. 초록색 상자가 감지될시 오른쪽 방향으로 보낸다.
5. 파란색 상자가 감지될시 왼쪽 방향으로 보낸다.
6. 상자를 보낸 후 약 3초후에 모터는 정지한다. 이는 컬러센서를 작동시킬 때 700ms 단위로 설정하였기 때문이며 RGBC를 4개의 값을 받아 약 3초정도의 시간이 소모된다.
7. 수동모드 선택 시 키패드에서 '5'를 기준으로 상하좌우로 모터의 방향을 조절할 수 있다. 2,8(상하) 4,6(좌우)를 사용한다.
8. 자동모드와 수동모드는 진입 때 뿐만 아니라 수시로 변경이 가능하다.

4.3 시스템 주요부분 설명(code)

본 프로젝트의 핵심이라고 하는 컬러센서에 관한 것과 어떻게 모터를 제어 하는지에 대해서만 설명하도록 하겠다. 컬러센서는 I2C 통신을 사용하는 이부분은 초음파 센서 실습을 진행했을 때와 동일하다. 하지만 컬러센서는 RGBC값은 16비트로 구성되어 있다. 그러니 각각의 데이터를 8비트씩 두 번 받아오는 과정이 필요하다. 위 과정에 대한 코드는 다음과 같다.

```
void TCS34725_GetRawData(unsigned int *red, unsigned int *green, unsigned int *blue, unsigned int *clear)
{
    unsigned char Htemp=0,Ltemp=0;

    TCS34725_read8(TCS34725_CDATAL,&Ltemp);
    TCS34725_read8(TCS34725_CDATAH,&Htemp);
    *clear=(Htemp*256) + Ltemp;

    TCS34725_read8(TCS34725_RDATAL,&Ltemp);
    TCS34725_read8(TCS34725_RDATAH,&Htemp);
    *red=(Htemp*256)+ Ltemp;

    TCS34725_read8(TCS34725_GDATAL,&Ltemp);
    TCS34725_read8(TCS34725_GDATAH,&Htemp);
    *green=(Htemp*256)+ Ltemp;

    TCS34725_read8(TCS34725_BDATAL,&Ltemp);
    TCS34725_read8(TCS34725_BDATAH,&Htemp);
    *blue=(Htemp*256)+ Ltemp;
    delay_ms(1000);
}
```

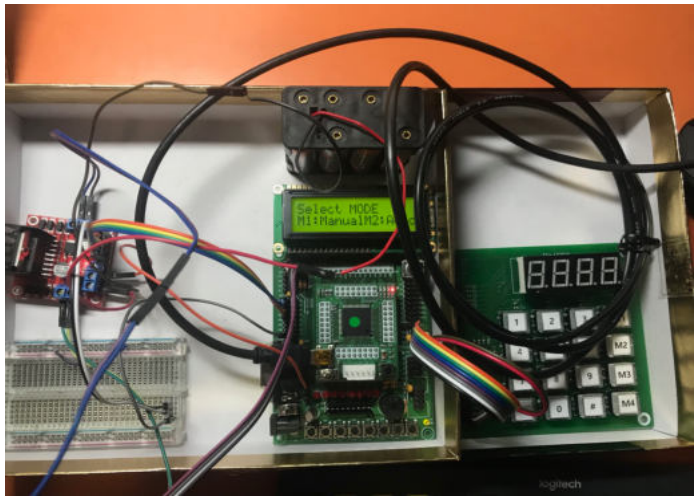
RGBC 에 대한 값들을 상위비트와 하위 비트를 구분해서 받아온다. 상위비트에 있는 데이터는 하위비트랑 같은 값이 아니라 8비트 만큼 큰 데이터 값이므로 256을 곱해주어서 받아왔다.

```
if (r>g && r>b && c>5000 && (m2==1 || m3==1)){
                PORTB.0 = 1;
                PORTB.1 = 0;
                PORTB.2 = 0;
                PORTB.3 = 0;
            }else if(g>r && g>b && c>5000 && (m2==1 || m3==1)){
                PORTB.0 = 0;
                PORTB.1 = 1;
                PORTB.2 = 0;
                PORTB.3 = 1;
            }else if(b>r && b>g && c>5000&& (m2==1 || m3==1)){
                PORTB.0 = 0;
                PORTB.1 = 1;
                PORTB.2 = 1;
                PORTB.3 = 0;
            }else if(c<5000 && (m2==1 || m3==1)){
                PORTB.0 = 0;
                PORTB.1 = 0;
                PORTB.2 = 0;
                PORTB.3 = 0;}
```

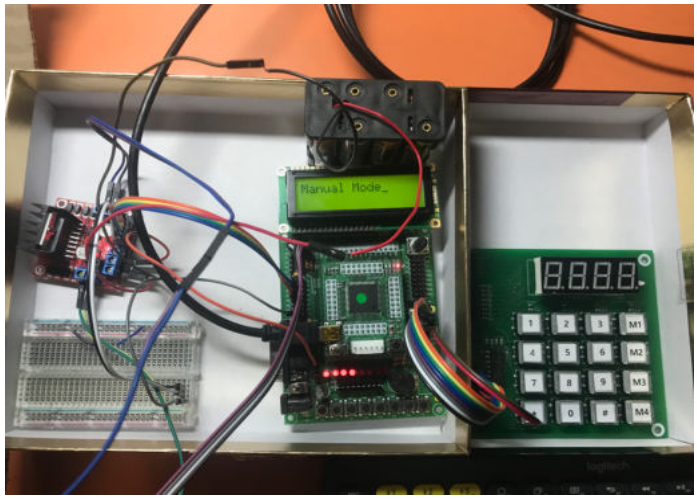
위 코드는 컬러센서 값에 따라 모터를 구동시키는 코드이다. 빛의 환경의 간섭을 많이 받아 정확한 data가 아닌 비교 값으로 설정하였으며 물체 유무는 Clear값을 활용하여 파악하였다.

5. 실험 및 고찰

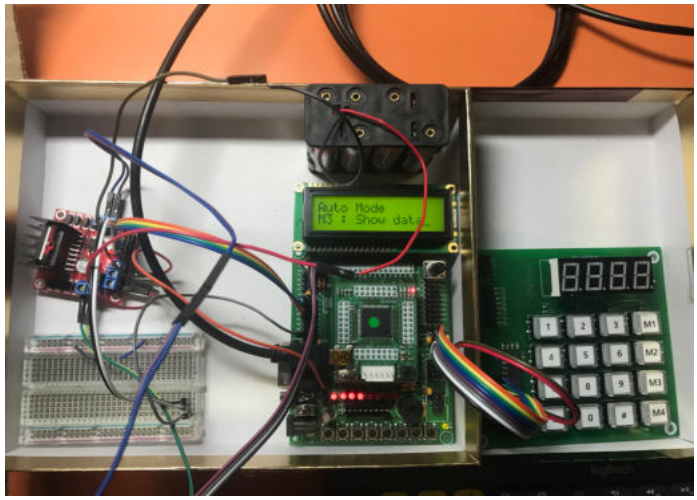
본 프로젝트의 실질적인 구동부인 컨베이어 벨트와 모터, 박스들의 모습이다. 제작은 사용한 택배상자를 재활용 하여 제작하였다. 컬러센서가 빛과 거리에 따라 다른 데이터값을 반환해 일정한 거리에 대한 환경을 만들어 주었다. 하지만 주변의 빛에 따른 환경 설정은 해주지 못하였다. 그럼에도 불구하도 위의 세가지 색상의 상자에 대해서 정확하게 구분 할 수 있도록 구현하였다.



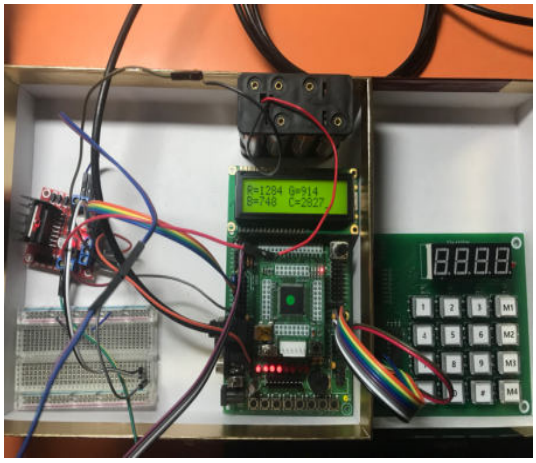
본 프로젝트의 조작부와 심장부의 모습이다. 전원은 1.5v배터리 8개를 직렬로 연결한 12v전원을 사용하였다. 현재 LCD에 표시되어 있는 화면은 초기 화면이며 모드를 선택하는 화면이다.



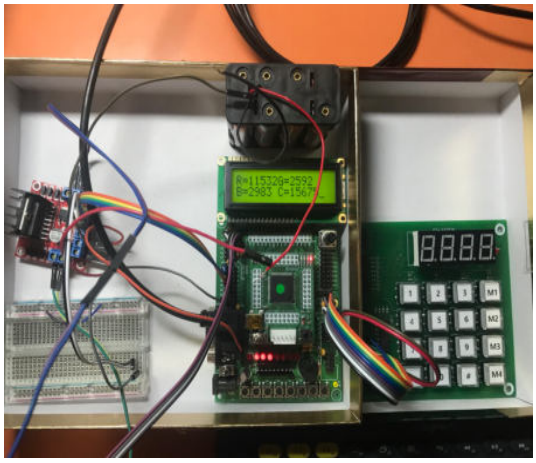
수동모드일때의 모습이다. 이때 키패드로 컨베이어 벨트를 조작 할 수 있다.



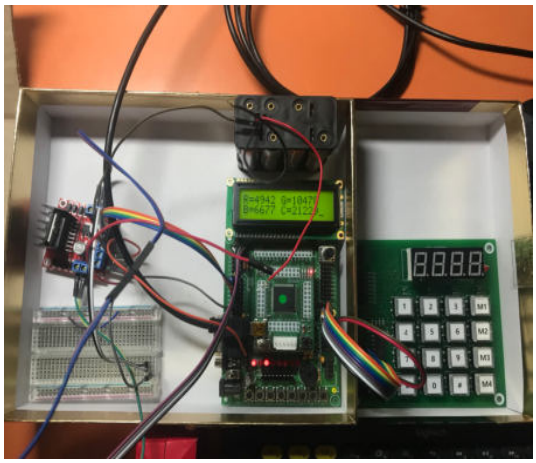
자동모드일때의 모습이다. 이때는 컬러센서의 값에 따라 컨베이어 벨트를 어떻게 구동할지 ATmega가 판단하여 작동한다. M3 버튼을 누르면 컬러센서의 데이터 값을 볼 수 있다.



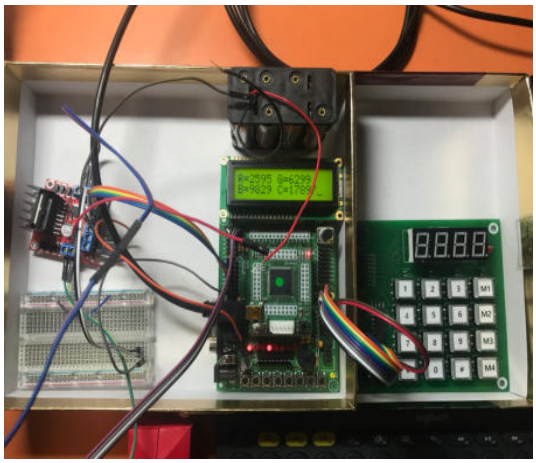
아무것도 상자가 없는 상태



빨간색 상자일 때



초록색 상자일 때



파란색 상자일 때

자동모드에서 M3 버튼을 눌러서 컬러센서의 데이터 값을 보는 모드로 진입하였다. 박스가 없을 때 RGB값도 작게 나오지만 Clear 값 또한 작게 나오는 모습을 확인할 수 있었다. 이를 이용해서 아무것도 없는 상황에 대한 해결을 할 수 있었다.
그 외의 색상값은 RGB 값을 이용하여 구분하였다.

6. 결론

6.1 시스템 구현 결과에 대한 논의

본 프로젝트의 결과에 대해서 가장 만족스러운 점은 초기 , 중간의 계획대로 프로젝트를 구현했다는 것이다. 특히 컬러센서를 제대로 사용하면서 가능한 일이었다. 하지만 컨베이어 벨트를 구현하는 과정에서 100% 온전한 작동이라고 보기는 어려운 결과가 나왔다. 정확하게 계산되지 않은 벨트의 길이, 지지대의 위치, 종이와 박스로 구현한 것, 1:1로 맞물려 돌지 않은 바퀴와 벨트 등이 원인이었던 것으로 판단된다. 그래서 실제 작동하는 모습을 보면 불안정한 모습을 많이 보인다. ATmega를 사용해 시스템을 작동시키는 것에는 성공하였지만 컨베이어 벨트를 구동하는 결과에 대해서는 부족한 결과이다.

6.2 학습 및 기대 효과

본 프로젝트를 진행하면서 초반부와 중반부에 기대되었던 기대효과에 대해서는 전부 완수하며 학습했다고 볼 수 있다. 기대되었던 기대효과는 다음과 같다.

1. 컬러센서의 작동방식에 대하여 학습하며 사용 할 수 있다.
2. 모터드라이버의 작동방식에 대하여 학습하며 사용 할 수 있다.
3. ATmega 의 통신에 대해여 학습하며 사용 할 수 있다.
4. 컨베이어 벨트의 작동 방식에 대하여 이해 할 수 있다.
5. 자동화 뿐 아니라 수동으로 컨베이어 벨트를 컨트롤 가능하다.
6. 물류센터의 자동화 분류에 대한 기초적인 알고리즘을 이해 할 수 있다.

위의 기대효과 이외에 Data sheet를 보는 능력을 본 프로젝트를 진행하기 전보다 향상되었다. 또한 프로토 타입의 하드웨어 제작이 소프트웨어를 짜는 것 만큼 쉬운 일이 아니라는 것을 학습 할 수 있었다.

6.3 향후 계획

본 프로젝트를 발전시켜서 실제 현장에서 사용하려면 많은 보완과 발전이 필요할 것으로 보여진다. 하지만 칼라센서의 모든 값들의 조합으로 물류를 분류 할 수 있다면 기존의 물류체제보다 더 발전된 형태의 물류분류 체계가 나올 수 있을 것이며 보다 많은 정보를 담아 더 다양한 기능을 가진 물류 시스템이 될 것이라 생각한다.