Implementation of Unmanned Ground Vehicle

MARIF MAHMOOD, ID-09221210

Department of Electrical and Electronics Engineering

APRIL 2010

BRAC University, Dhaka, Bangladesh

Declaration

I hereby declare that this thesis is based on the results found by our group. Materials found by other researchers are mentioned at the reference. This thesis, neither in part or in whole , has been previously submitted for any degree.

15/4/10

Signature of the Supervisor

Masif Mahmood

Signature of the Author

Acknowledgement

I would like to express our deep gratitude to many people who helped and encouraged us throughout the time of our project.

First and foremost, I would like to express our heartiest gratitude to our supervisor Dr. Azad AKM and our co-supervisor Mehdi Hasan Sadi. I am in debt to them for their support, guidance and the valuable impact they left on our approach on this project.

This research had been performed under the Electrical and Electronics Engineering Department of BRAC University. I appreciate the attempts by the department to improve the attitudes toward gradual education and research.

BRAC University, 2010

Abstract

Unmanned ground vehicles (UGV) are robotic platforms that are used as an extension of human capability. This type of robot is generally capable of operating outdoors and over a wide variety of terrain, functioning in place of humans. UGVs have counterparts in aerial warfare (unmanned aerial vehicle) and naval warfare (remotely operated underwater vehicles). Unmanned robotics is actively being developed for both civilian and military use to perform dull, dirty, and dangerous activities. There are two general classes of unmanned ground vehicles: Teleoperated and Autonomous. A teleoperated UGV is a vehicle that is controlled by a human operator at a remote location via a communications link. All cognitive processes are provided by the operator based upon sensory feedback from either line-of-sight visual observation or remote sensory input such as video cameras. A basic example of the principles of teleoperation would be a toy remote control car. An autonomous UGV is essentially an autonomous robot but is specifically a vehicle that operates on the surface of the ground. Applications of UGV range from remote surveillance to scattered mine detection. The purpose of the thesis is to design a teleoperated unmanned vehicle mounted with a camera. The vehicle will be initially controlled via wire. Images sent by the camera will be used for navigating the vehicle to the desired location. User sitting in front of an end device such as a computer would be receiving and viewing images sent by the camera and based on that would steer the vehicle. In further development the vehicle will be controlled by Radio Frequency from a distant location.

Title of Contents

CHAPTER 1

1.1 Introduction and Motivation

Unmanned Ground Vehicles (UGV) are robotic platforms that are used as an extension of human capability. This type of robot is generally capable of operating outdoors and over a wide variety of terrain, functioning in place of humans. UGVs have counterparts in aerial warfare (unmanned aerial vehicle) and naval warfare (remotely operated underwater vehicles). Unmanned robotics is actively being developed for both civilian and military use to perform dull, dirty, and dangerous activities. There are two general classes of unmanned ground vehicles: Tele-operated and Autonomous. A tele-operated UGV is a vehicle that is controlled by a human operator at a remote location via a communications link. All cognitive processes are provided by the operator based upon sensory feedback from either line-of-sight visual observation or remote sensory input such as video cameras. A basic example of the principles of tele-operation would be a toy remote control car. An autonomous UGV is essentially an autonomous robot but is specifically a vehicle that operates on the surface of the ground. Applications of UGV range from remote surveillance to scattered mine detection.

It is expected that groups of unmanned ground and air vehicles will play a major role in future civilian and military applications. Algorithms for their coordination and control must account for the dynamic nature of the environment in which they will operate. With this capability, robots can be used in applications including search and rescue, exploration, surveillance, and scientific data gathering. In this project, we are attempting to design a practical framework for online controlling of an unmanned ground vehicle. The vehicle will be supervisory controlled. The images sent back by the camera will assist in such purposes.

The purpose of the thesis is to design a tele-operated unmanned vehicle mounted with a camera. The vehicle will be initially controlled via wire. Images sent by the camera will be used for navigating the vehicle to the desired direction. User sitting in front of an end device such as a computer would be receiving and viewing images sent by the camera and based on that would steer the vehicle. Once the vehicle can be controlled using the wired connection further development, if possible within the semester's time, will be attempted.

The following chapters give a detailed view of the entire project.

Chapter2 describes how the entire vehicle is going to be supervised based on the feedback of the webcam.

Chapter3 is about the micro-controller used and the functioning of the micro-controller downloader.

Chapter 4 is about the system overview about how the system is being marshaled.

Chapter 5 is about the drivers that are being used along with the motors.

Chapter 6 is about the wireless communication of the UGV.

Chapter 7 describes the applications of UGV.

Chapter 8 describes the conclusion and the future works of the project.

CHAPTER 2

2.1 Project overview

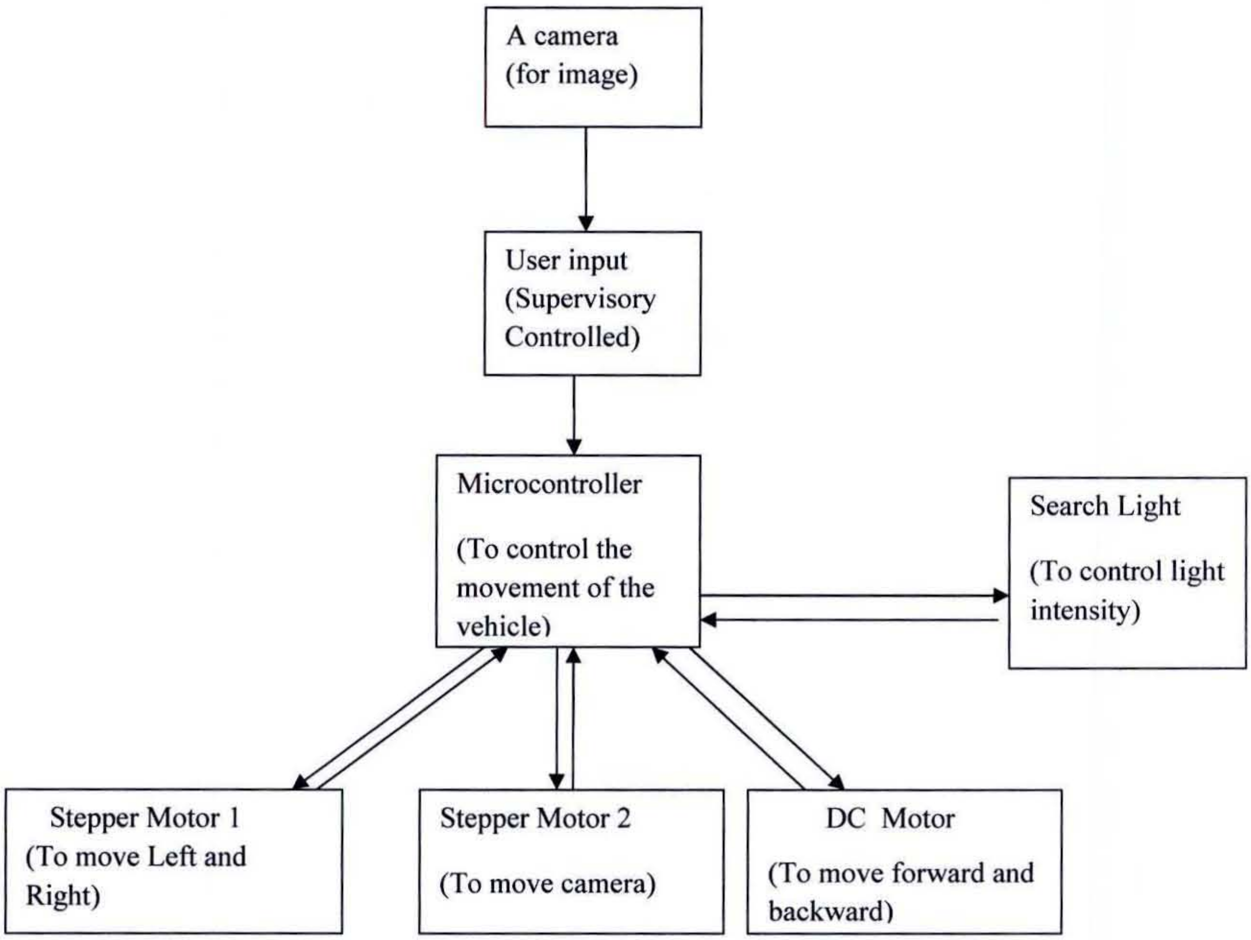


Figure 1: System overview

The project was carried out with the intention of constructing a vehicle that will be supervisory controlled. A web-cam of resolution 5 Mega pixel was used to collect images and send to the supervisory computer. The user sends input from the keyboard and these signals are then processed in to the micro-controller. There are a total of 8 inputs and 18 outputs defined on the micro-controller. The micro-controller is programmed to operate a pair of stepper motors, a DC motor and a search light. The purpose of the stepper motors are for rotation while the DC motor is being used for the forward and backward movement of the vehicle. The search light was used for illuminating dark places thus facilitating the webcam to take clear images.

CHAPTER 3

3.1 Micro-controller

The micro-controller used in this system is AT MEGA 32 of the ATMEL AVR family. The ATMEL AVR FAMILY has certain features that make it suitable for the system. The features are being:

1. RISC architecture with mostly fixed-length instruction, load-store memory access, and 32 general-purpose registers.
2. A two-stage instruction pipeline that speeds up execution.
3. Majority of instructions take one clock cycle.
4. Up to 10-MHz clock operation.
5. Wide variety of on-chip peripherals, including digital I/O, ADC, EEPROM, Timer, UART, RTC timer, pulse width modulator (PWM), etc.
6. Internal program and data memory.
7. In-system programmable.
8. Available in 8-pin to 64-pin package size to suit wide variety of applications.
9. Up to 12 times performance speedup over conventional CISC controllers.
10 Wide operating voltage from 2.7 V to 6.0 V.
11. A simple architecture offers a small learning curve to the uninitiated.

The specifications of AT MEGA 32 are being:

1. High-performance, Low-power AVR® 8-bit Microcontroller
2. Advanced RISC Architecture
– 131 Powerful Instructions – Most Single-clock Cycle Execution
– 32 x 8 General Purpose Working Registers
– Fully Static Operation
– Up to 16 MIPS Throughput at 16 MHz
– On-chip 2-cycle Multiplier

1. Nonvolatile Program and Data Memories

– 32K Bytes of In-System Self-Programmable Flash

Endurance: 10,000 Write/Erase Cycles

– Optional Boot Code Section with Independent Lock Bits

In-System Programming by On-chip Boot Program

True Read-While-Write Operation

– 1024 Bytes EEPROM

Endurance: 100,000 Write/Erase Cycles

– 2K Byte Internal SRAM

– Programming Lock for Software Security

4. JTAG (IEEE std. 1149.1 Compliant) Interface

– Boundary-scan Capabilities According to the JTAG Standard

– Extensive On-chip Debug Support

– Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

5. Peripheral Features

– Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes

– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

– Real Time Counter with Separate Oscillator

– Four PWM Channels

– 8-channel, 10-bit ADC

8 Single-ended Channels

7 Differential Channels in TQFP Package Only

2 Differential Channels with Programmable Gain at 1x, 10x, or 200x

– Byte-oriented Two-wire Serial Interface

– Programmable Serial USART

– Master/Slave SPI Serial Interface

– Programmable Watchdog Timer with Separate On-chip Oscillator

– On-chip Analog Comparator

6. Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection

– Internal Calibrated RC Oscillator

– External and Internal Interrupt Sources

– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby

7. I/O and Packages

– 32 Programmable I/O Lines

– 40-pin PDIP, 44-lead TQFP, and 44-pad MLF

8. Operating Voltages

– 2.7 - 5.5V for ATmega32L

– 4.5 - 5.5V for ATmega32

9. Speed Grades

– 0 - 8 MHz for ATmega32L

– 0 - 16 MHz for ATmega32

10. Power Consumption at 1 MHz, 3V, 25°C for ATmega32L

– Active: 1.1 mA

– Idle Mode: 0.35 mA

– Power-down Mode: < 1 μA

The AVR line provides a full range of processing power, from small 8-pin processors to complex 100-pin processors. The same compiler and programming hardware may be used with a wide variety of microcontrollers. Many of the AVR microcontrollers are available in dual inline package, which makes them readily useable on a printed circuit board prototype (e.g., senior design projects). Many of the microcontrollers in the AVR line are pin-for-pin compatible with one another. This allows easy move up and down the AVR line as the project becomes better defined.

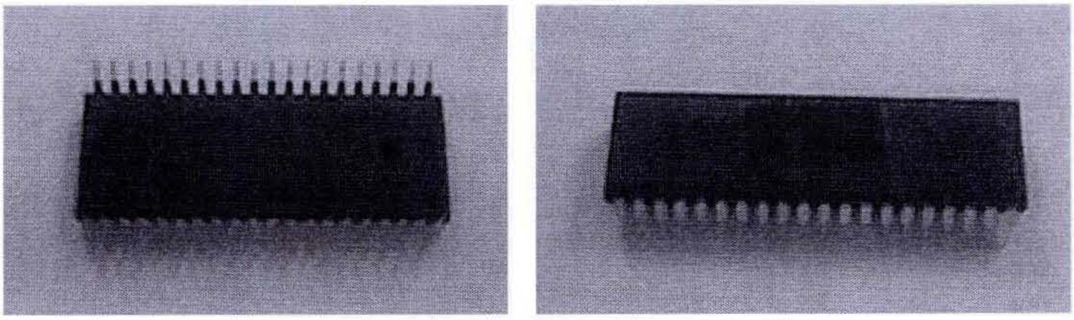


Figure 2: Micro-controller

3.2 Micro-controller Downloader

The pre-thesis work consisted of making a micro-controller downloader. The work was divided into two stages. At the first stage the downloader was constructed on the Vero-board using the circuit as shown in figure 2.

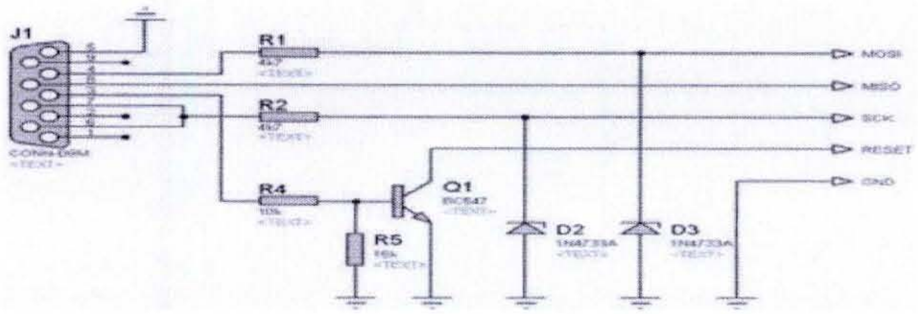


Figure 3- Circuit diagram

Once the circuit was in operation, the downloader was constructed on a double layered copper board. The circuit was designed using the software PROTEUS v6.9. This was developed into a Printed Circuit Board. The making of this PCB was the second phase of the pre-thesis work. Figure 3 illustrate the circuit that was developed using the PROTEUS software.

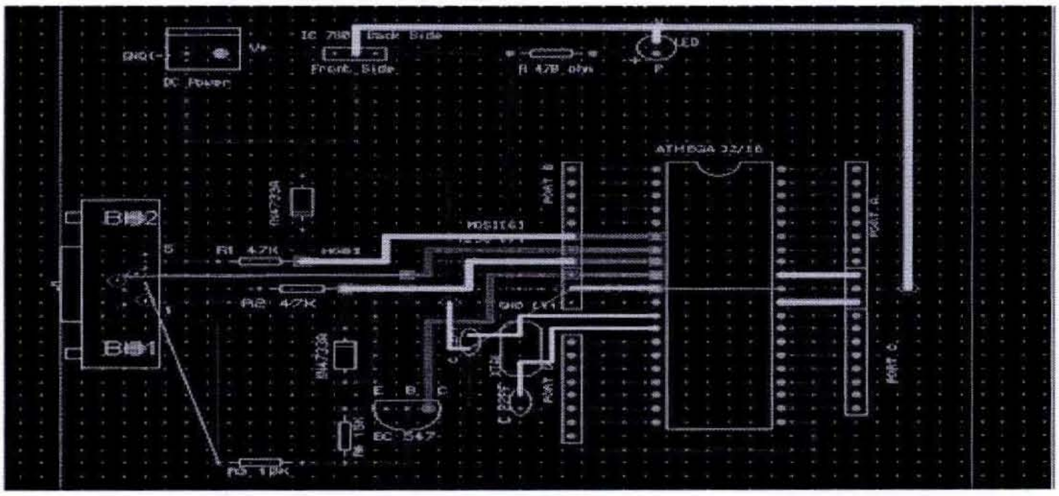


Figure 4- Circuit designed in PROTEUS v6.9

Layouts of the designed circuit were then printed. Figure5 illustrates the layouts of the circuit.

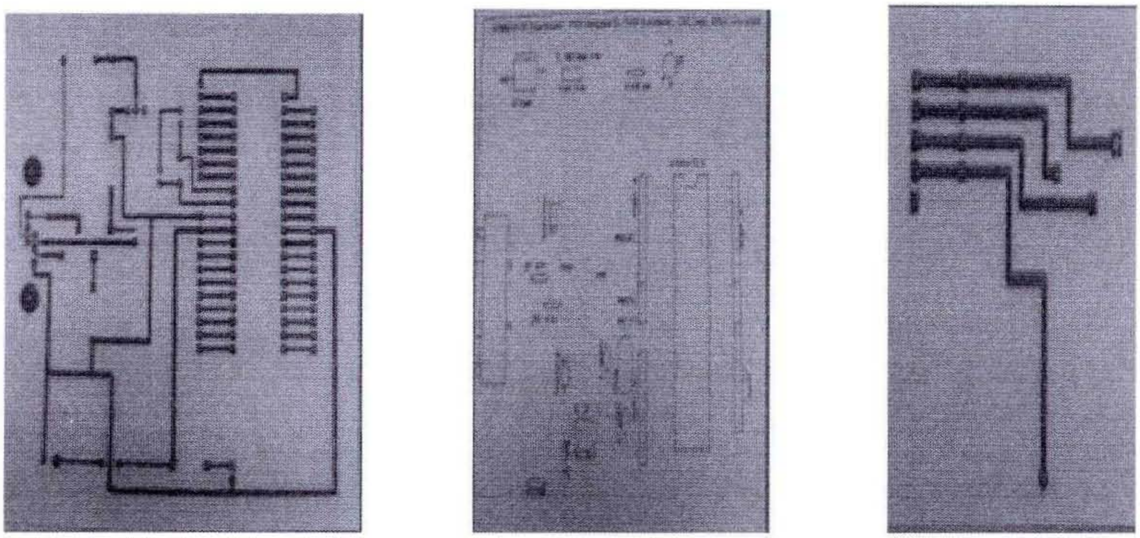


Figure-5 – Circuit layout.

The layouts developed were then impinged on a double layered copper board. Figure 5 illustrates the board which is otherwise known as Printed Circuit Board.

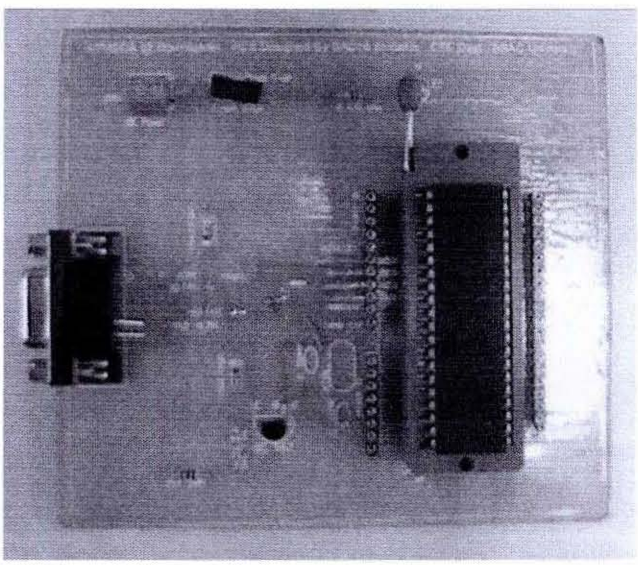


Figure 6- Printed Circuit Board.

CHAPTER 4

4.1 Inputs from keyboard

The vehicle constructed is coordinated by the usage of keyboard inputs. The camera placed at the very front end of the vehicle captures image and sends to the computer to which it is connected. A length of the wire of the webcam is a major limitation as it restricts the distance over which the vehicle can be moved since the early part of the project involves wired communication. To overcome this limitation an extender of length 9cm was connected to the webcam. The images sent by the camera were a live feedback. Based on this feedback the user at the computer end decides to steer the vehicle in the appropriate direction. The central processing unit of the computer (CPU) was connected to the vehicle by means of a 25 pin parallel port connection. The latter also used an extender cable to overcome the limitation of the short length wire. Pin numbered 2 to 9 were used for data transmission while the pins numbered from 18 to 25 were grounded. The data pins were connected to the input pins of the micro-controller. The data pins acted as the inputs. The conveyed the input signals from the keyboard.

A major problem of the 25 pin parallel port was the output voltage at the receiving end. The voltage obtained at the high logic ("1") was 3.25V. However this was not enough as the minimum operating voltage of the micro-controller AT MEGA32 is being 4.5V. Another problem associated with the system was being the external power supply. An external power supply of 12V was used to drive the motors of the vehicle. As such there was a chance of power surging back into the CPU and cause irreparable damage. To prevent aforementioned problems an opto-isolator was used. The latter acted as a buffer and also prevented back surge of power.

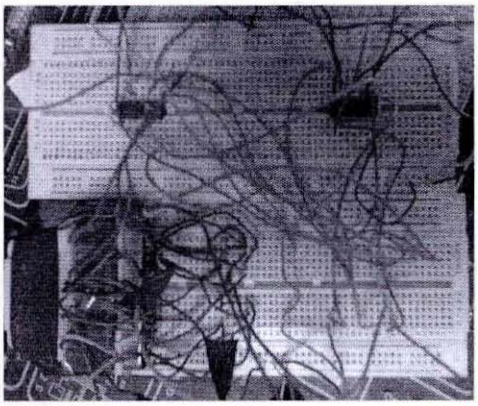


Figure 7 – Parallel port inputs

4.2 Forward and backward motion

The forward and the reverse direction of motion of the vehicle were controlled using a DC motor. Inputs from the keyboard were fed to the Port C of the micro-controller. The corresponding outputs were passed to the H-bridge and then to the DC motor. The DC motor had an operating voltage of 12V and a high current rating. Hence H bridge was used to adjust the current rating and also to change the direction of motion of the vehicle. The MOSFETS used while implementing the H bridge were IRF 530. The 12V needed to operate the vehicle was fed directly to the H bridge. The speed of the DC motor was set at three different levels- slow, medium and fast. The technique used for this speed control was Pulse Width Modulation (PWM). In PWM the duty cycle is varied and therefore the duration for impulses which is generally considered as the high logic ("1") is also varied. So the DC motor receives a set of impulses with different pulse duration. Since three different speed levels was defined, the DC motor was fed with three different pulse. The pulse having the lower duration was assigned as the slow while the pulse having the higher duration was assigned as the fast. The micro-controller had inputs defined at different combinations for speed control.

4.3 Front Wheel Steering

The front wheel is controlled using a stepper motor of resolution 1.8°. The operating voltage of the motor was 2.6V. Two rechargeable batteries of total power 12.6V were used to supply the required voltage. The stepper motor had a high current rating and so required additional circuitry. Four darlington pairs were used to supply the current. The inputs of the darlington pair were from the micro-controller and the corresponding outputs were connected to the stepper motor.

The motor was programmed to rotate a maximum angle of 36° and a minimum angle of 3.6°. Three different speed levels were also assigned for this particular motor. The speed was varied by varying the rate of pulses applied at the input end of the micro-controller.

4.4 Web-cam Rotation

A web-cam of resolution 5 mega pixels is used for collecting images and sending live feedback to the CPU. The purpose of the web-cam is for navigation. Hence it is placed at the front end of the vehicle on top of a stepper motor. The stepper motor is identical to the one used for steering the front wheel. This stepper motor too had a darlington pair to adjust the current rating. The stepper motor is also set to rotate at three different speeds in two directions, clockwise and counter-clockwise.

4.5 Search Light Control

The web-cam used assists in navigation However in a dark zone the feedbacks from the web-cam was difficult to decipher. Hence a search light was designed to enable the web-cam to operate effectively in the dark. The web-cam was constructed using a number of LEDs. The LEDs were connected in groups so that on receiving pulse 3 LEDs start glowing at a time. A total of 9 LEDs were used to construct the search light.

To control the light intensity of the search light, the LEDs were connected in groups. On receiving the first impulse, the first group comprising of three LEDs would turn on. Successive pulse would then turn on the remaining LEDs.

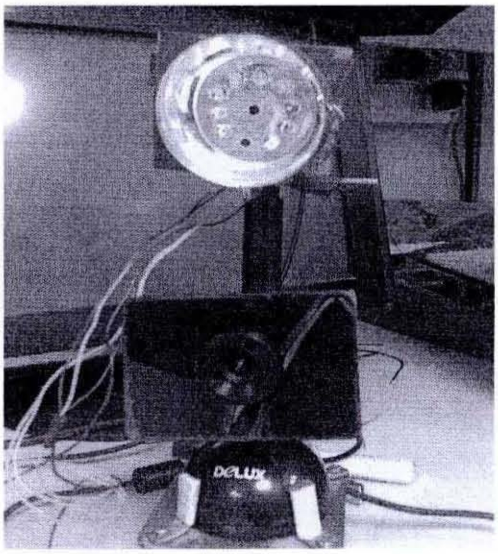


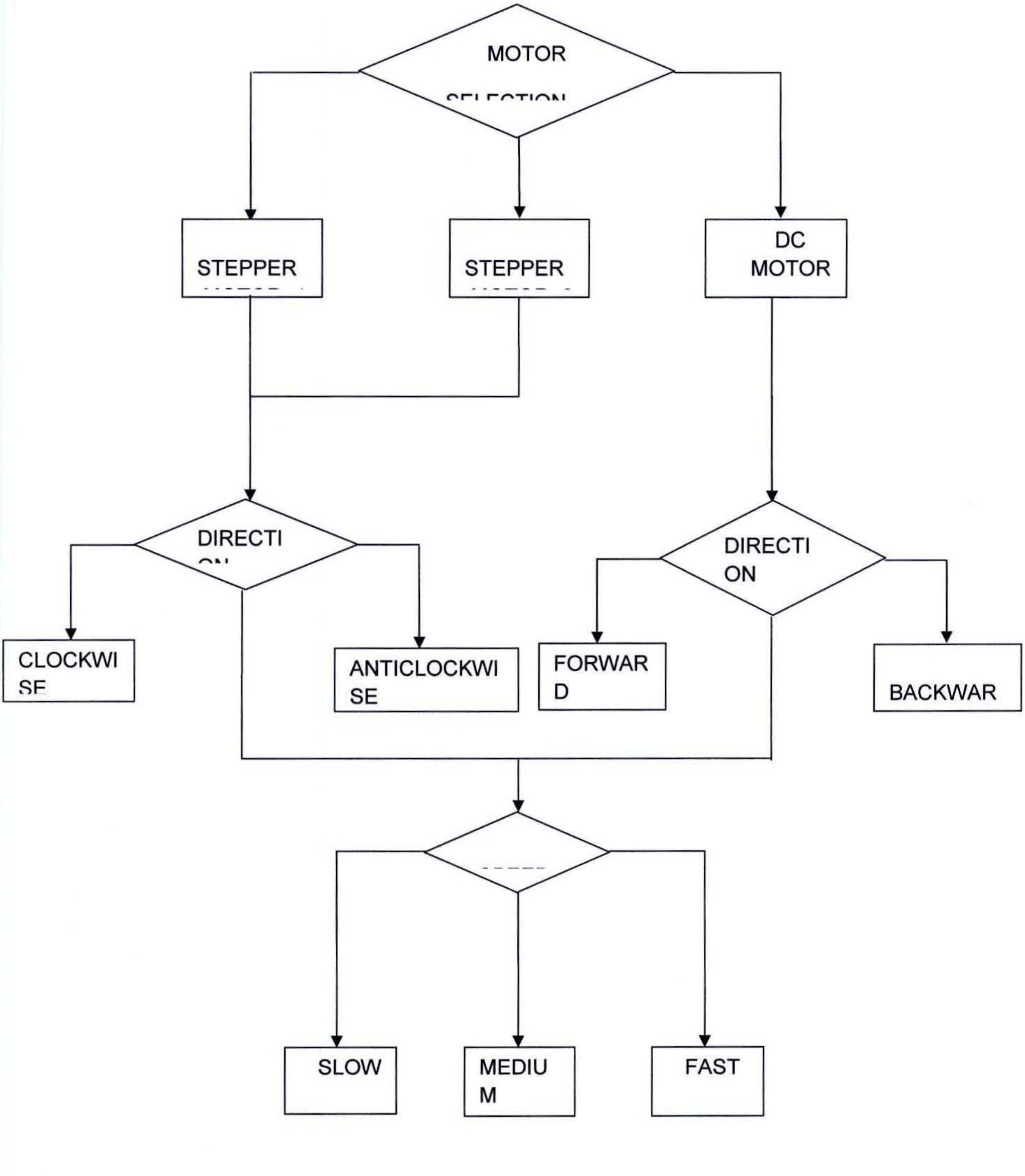
Figure 8– Search light and the web-cam

4.6 Power management

The vehicle is powered by a pair of rechargeable batteries. Each battery has a voltage rating of 6.3V. The batteries are connected in series to supply a total voltage 12.6V. The micro-controller requires 5V to operate. As such a voltage regulator is used to supply 5V. The voltage regulator is connected directly to the battery. The DC motor is directly connected to the battery as it requires 12V of its own.

Figure 9–Power Supply

4.7 System Overview



CHAPTER 5

DRIVERS

5.1 Stepper Motor Driver

a) H-bridge

For our project we used solid state H-bridge. A solid-state H-bridge is typically constructed using reverse polarity devices (i.e. BJTs ,PNP or P-channel MOSFETs connected to the high voltage bus and NPN BJTs or N-channel MOSFETs connected to the low voltage bus).

The most efficient MOSFET designs use N-channel MOSFETs on both the high side and low side because they typically have a third of the ON resistance of P-channel MOSFETs. This requires a more complex design since the gates of the high side MOSFETs must be driven positive with respect to the DC supply rail. However, many integrated circuit MOSFET drivers include a charge pump within the device to achieve this.

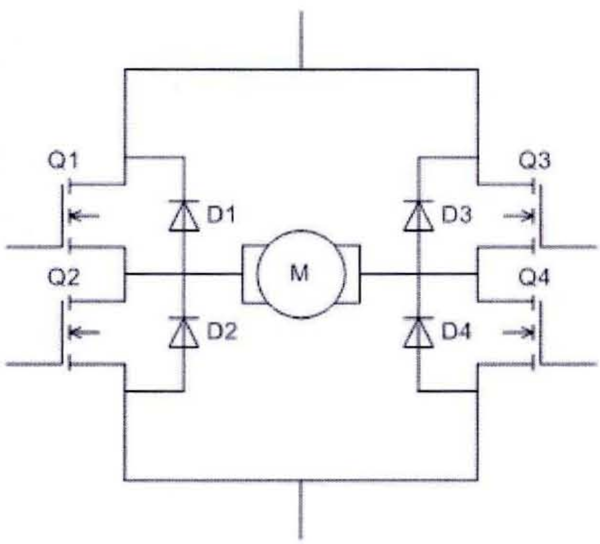


Figure 10- Solid state H-bridge

Alternatively, a switch-mode DC-DC converter can be used to provide isolated ('floating') supplies to the gate drive circuitry. A multiple-output fly back converter is well-suited to this application.

Another method for driving MOSFET-bridges is the use of a specialized transformer known as a GDT (Gate Drive Transformer), which gives the isolated outputs for driving the upper FETs gates. The transformer core is usually a ferrite toroid, with 1:1 or 4:9 winding ratio. However, this method can only be used with high frequency signals. The design of the transformer is also very important, as the leakage inductance should be minimized, or cross conduction may occur. The outputs of the transformer also need to be usually clamped by zener diode because high voltage spikes could destroy the MOSFET gates.

A common variation of this circuit uses just the two transistors on one side of the load, similar to a class AB amplifier. Such a configuration is called a "half bridge". The half bridge is used in some switched mode power supplies that use synchronous rectifiers and in switching amplifiers. The half H-bridge type is commonly abbreviated to "Half-H" to distinguish it from full ("Full-H") H-bridges. Another common variation, adding a third 'leg' to the bridge, creates a 3-phase inverter. The 3-phase inverter is the core of any AC motor drive.

A further variation is the half-controlled bridge, where one of the high- and low-side switching devices (on opposite sides of the bridge) are replaced with diodes. This eliminates the shoot-through failure mode, and is commonly used to drive variable/switched reluctance machines and actuators where bi-directional current flow is not required. A "double pole double throw" relay can generally achieve the same electrical functionality as an H-bridge (considering the usual function of the device). An H-bridge would be preferable to the relay where a smaller physical size, high speed switching, or low driving voltage is needed, or where the wearing out of mechanical parts is undesirable.

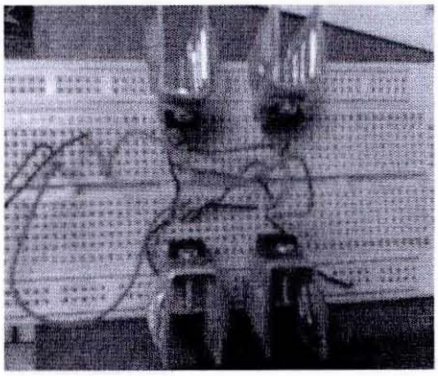


Figure 11- H-bridge

5.2 Stepper Motor Driver

Darlington Pair

The Darlington transistor (often called a Darlington pair) is a compound structure consisting of two bipolar transistors (either integrated or separated devices) connected in such a way that the current amplified by the first transistor is amplified further by the second one This configuration gives a much higher current gain (written β, h_{fe}, or h_{FE}) than each transistor taken separately and, in the case of integrated devices, can take less space than two individual transistors because they can use a shared collector.

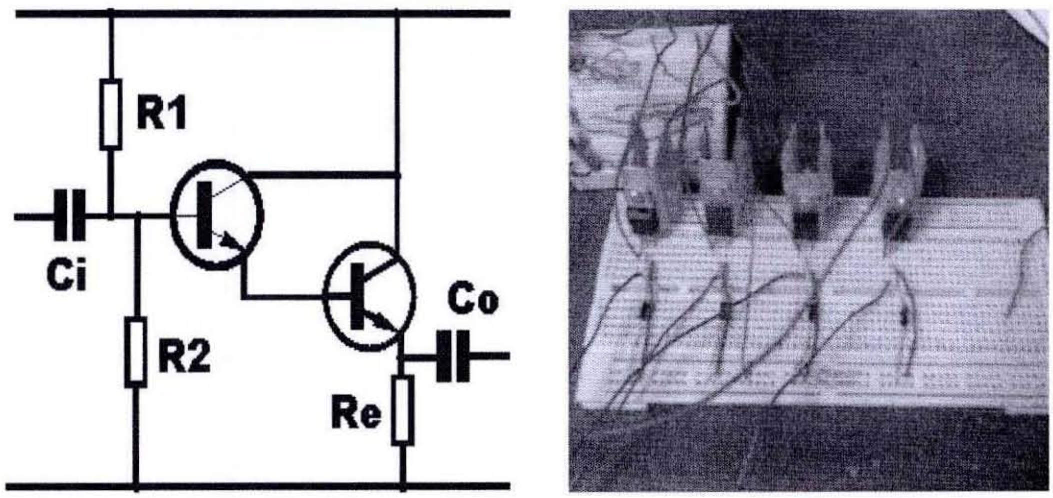


Figure 12- Darlington Pair

5.3 Buffer

The vehicle is being steered by the usage of the keyboard. The input signals are conveyed to the micro-controller through a parallel port. At the output of the parallel port a maximum voltage of 3.25V is obtained. However a micro-controller operates at a voltage of 5V. Hence the need of a buffer. Initially the buffer was constructed using two inverters. The inverters are connected back to back to amplify the 3.25V to 5V. Further modification was then made to the buffer. The buffer was developed using the IC 74126. This IC belongs to the LS series. It is otherwise known as QUAD 3-STATE BUFFER. This buffer had a disadvantage of its own and it being the fact that the IC requires 6.3V to show 5.0V at its output. Hence the first buffer constructed using cascaded inverters is used. The buffer is placed between the parallel port and the micro-controller. The 8 pins of the parallel port that transmits the data from the keyboard connect to the 74126. The outputs of the IC then converge to the micro-controller.

5.4 Opto-isolator

An opto-isolator is a device that uses a short optical transmission path to transfer an electronic signal between elements of a circuit, typically a transmitter and a receiver, while keeping them electrically isolated—since the electrical signal is converted to a light beam, transferred, then converted back to an electrical signal, there is no need for electrical connection between the source and destination circuits. For the project the IC QTC 4N35 9826K was being used.

The opto-isolator is simply a package that contains both an infrared light-emitting diode (LED) and a photodetector such as a photosensitive silicon diode, transistor Darlington pair, or silicon controlled rectifier (SCR). The wave-length responses of the two devices are tailored to be as identical as possible to permit the highest measure of coupling possible. Other circuitry—for example an output amplifier—may be integrated into the package. An opto-isolator is usually thought of as a single integrated package, but opto-isolation can also be achieved by using separate devices.

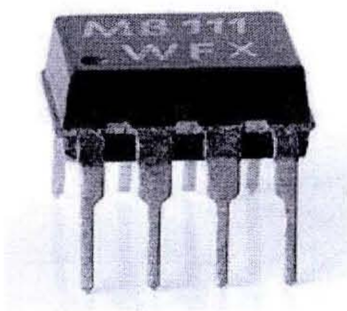


Figure 13 Opto-isolator

A common implementation is a LED and a phototransistor in a light-tight housing to exclude ambient light and without common electrical connection, positioned so that light from the LED will impinge on the photodetector. When an electrical signal is applied to the input of the opto-isolator, its LED lights and illuminates the photodetector, producing a corresponding electrical signal in the output circuit. Unlike a transformer the opto-isolator allows DC coupling and can provide any desired degree of electrical isolation and protection from serious overvoltage conditions in one circuit affecting the other. A higher transmission ratio can be obtained by using a Darlington instead of a simple phototransistor, at the cost of reduced noise immunity and higher delay

With a photodiode as the detector, the output current is proportional to the intensity of incident light supplied by the emitter. The diode can be used in a photovoltaic mode or a photoconductive mode. In photovoltaic mode, the diode acts as a current source in parallel with a forward-biased diode. The output current and voltage are dependent on the load impedance and light intensity. In photoconductive mode, the diode is connected to a supply voltage, and the magnitude of the current conducted is directly proportional to the intensity of light. This optocoupler type is significantly faster than photo transistor type, but the transmission ratio is very low; it is common to integrate an output amplifier circuit into the same package.

The optical path may be air or a dielectric waveguide. When high noise immunity is required an optical conductive shield can be integrated into the optical path. The transmitting and receiving elements of an optical isolator may be contained within a single compact module, for mounting, for example, on a circuit board; in this case, the module is often called an **optoisolator** or **opto-isolator**. The photosensor may be a photocell, phototransistor, or an optically triggered SCR or TRIAC. This device may in turn operate a power relay or contactor.

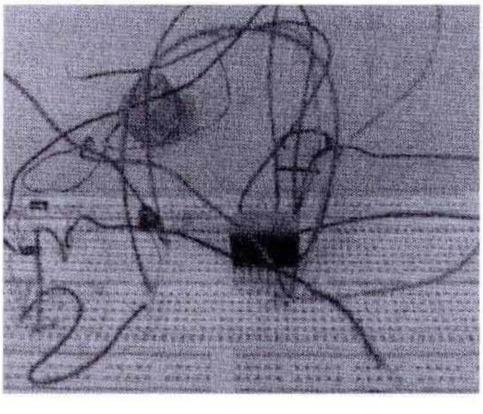


Figure 14 Opto-isolator

5.5 Relay circuit

A relay is an electrically operated switch. Many relays use an electromagnet to operate a switching mechanism, but other operating principles are also used. Relays find applications where it is necessary to control a circuit by a low-power signal, or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits, repeating the signal coming in from one circuit and re-transmitting it to another. Relays found extensive use in telephone exchanges and early computers to perform logical operations. A type of relay that can handle the high power required to directly drive an electric motor is called a contactor. Solid-state relays control power circuits with no moving parts, instead using a semiconductor device triggered by light to perform switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults; in modern electric power systems these functions are performed by digital instruments still called "protection relays".

For the project, a relay similar to the one shown in the figure below was constructed.

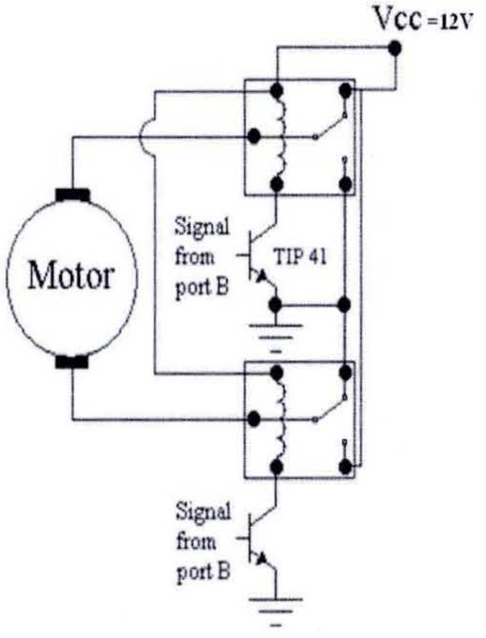


Figure 15- Relay Circuit

CHAPTER 6

WIRELESS COMMUNIACTION

6.0 Wireless Communication

A wireless communication module was built on the vehicle in order to overcome the limitation imposed by the wired communication. The main limitation in the wired communication was the distance that the vehicle could traverse. The length of the wire was the maximum distance that the vehicle could move. A wireless communication between the control unit and the vehicle would help in increasing the range of the vehicle. The wireless communication module was built using KST-Tx01, the transmitter, and KST-Rx01. The mode of communication was point to point. Amplitude Shift Keying was used during the process of modulation. To enhance the communication between the transmitter and the receiver Manchester coding was being integrated with the programming. The maximum range of the transmitter was 15000m which more than fulfilled our requirements. The specifications of the receiver and the transmitter are given below:

1. 1000M launch distance
2. 3-5V mi-ni transmitter
3. 3.Various frequency available
4. Anti-disturbing
5. Consistency better
6. Transmit power:1W
7. Operating frequency: 315MHz~433.92MHz
8. Operating Temp: 40~80
9. Operating Voltage : 3V~5V
10. Modulation Type: ASK

CHAPTER 7

APPLICATIONS OF UGV.

7.0 Applications of UGV

The vehicle constructed can be controlled from a remote location. It is a supervisory controlled vehicle. This vehicle is mounted with a webcam to enable it to collect image and send back feedback. This vehicle has a number of applications. The applications are being:

a) Deep sea exploration.

b) Tunnels that have very narrow passage and low illumination can be explored.

c) Rescue purpose- to look for survivors in a disaster struck area.

d) Exploration of volcanoes.

e) Remote surveillance.

f) Counter Intelligence.

g) Mobile check-points.

CHAPTER 8

8.1 Conclusion

The vehicle was designed to work acquiring data from the feedback sent by the webcam. The system would run according to the actual algorithms described in the respective section. However during the test sessions factors like current rating of the motors, frictional force, displacement of the front wheel etc affected the motion of the vehicle. The length of the wired webcam hindered the motion of the vehicle. It also limited the distance over which the vehicle can be moved.

8.2 Future Work

Thus, the system needs to be worked upon to remove the limitations imposed by the wire connections. It can be improved by using a wireless webcam. The vehicle can also be mounted with a heat sensor so that it can be used in rescue purposes.

REFERENCES

[1] Wikipedia. (2008). Decca Navigator System [online]. [Accessed 13

September 2008]. Available from World Wide

Web: <http://en.wikipedia.org/wiki/Decca_Navigator_System>.

[2] Raphael Ahmed, Md. Intekhabul Hafiz, *UNMANNED VEHICLE NAVIGATION USING GLOBAL POSITIONING SYSTEM (GPS).*

[3] http://www.ai.sri.com/people/flakey/control.html.

[4] Madhavan, R.; Schlenoff, C. *The effect of process models on short-term prediction of moving objects for unmanned ground vehicles.*

[5] Yu Jun; Ma Di; Meng Joo Er;. *Development of a fuzzy controller based mini autonomous unmanned ground vehicle.*

[6] Hee Chang Moon; Kyoung Moo Min; Jung Ha Kim; *Vision system of Unmanned Ground Vehicle.*

[7] Sang Jin Lee; Dong Myung Lee; Jae Cheon Lee; *Development of communication framework for unmanned ground vehicle.*

APPENDICES

APPENDICEX A

CLOCKWISE ROTATION OF THE STEPPER MOTOR.

```
#include<avr/io.h>
#include<util/delay.h>
#include <stdlib.h>
int main(){
        unsigned char p;
        int n;
    DDRD=0xFB;      //portD as input
          PORTB=0x00;

     DDRB=0xFF;

     p= 0b00001000;
               PORTB = 0b11111111;
               n=1;
     while(1)
     {
 if(PIND & 0b00000100)
        {
              _delay_ms(100.0);
           PORTB=p;
```

```
            p=p>>1;
        n=n+1;
       if(n==5)
       {
      p= 0b00001000;
                    n=1;
   }
   }


     }
   return 0;
}
```

ANTI-CLOCKWISE ROTATION OF THE STEPPER MOTOR.

```
#include<avr/io.h>
#include<util/delay.h>
#include <stdlib.h>
int reverse(){
        unsigned char p;
        int n;


    DDRD=0xFB;       //portD as input
          PORTB=0x00;
```

```
DDRB=0xFF;

p= 0b01000000;

PORTB = 0b11111111;

n=1;

while(1)


{

if(PIND & 0b00100000)

{

_delay_ms(500);

PORTB=p;


p=p<<1

n=n+1;

if(n==5)

{

p= 0b00100000;

n=1;

}
```

```
}

}

return 0;

}
```

APPENDICEX B

FINAL PROGRAM

```
#define STOP_MOTOR TCCR1B = 0x00; TCCR1A = 0x00

#define START_MOTOR TCCR1B = 0x09

#define set_FORWARD TCCR1A = 0x81

#define set_REVERSE TCCR1A = 0x21


#define sbi(address,bit) (address |= (1<<bit));

#define cbi(address,bit) (address &=~(1<<bit));


#define OUTPUT_PORT PORTB

#define OUTPUT_PORT1 PORTA


#define COUNTER_LOWER_LIMIT 0x0000

#define COUNTER_UPPER_LIMIT 0x00f8

#define STEP_SIZE 0x0008


#define LED1 PB0

#define LED2 PB1

#define LED3 PB2

#define LED4 PB3


#include<avr/io.h>

#include<util/delay.h>
```

```
#include<avr/interrupt.h>

#define F_CPU 8000000UL

void delay_ms(unsigned int ms){

   while(ms){
          _delay_ms(1.000);
          ms--;
          }
}

int main()
{



    DDRD |= (1<<PD5) | (1<<PD4);
          DDRB |= (1<<PB0) | (1<<PB1) | (1<<PB2) | (1<<PB3);
          DDRB |= (1<<PB7) | (1<<PB6) | (1<<PB5) | (1<<PB4);
          DDRA |= (1<<PA7) | (1<<PA6) | (1<<PA5) | (1<<PA4) | (1<<PA3) | (1<<PA2) | (1<<PA1) |
(1<<PA0);;
          PORTC = 0x00ff;
          DDRC = 0x0000;
          cbi(DDRD,PD0);
          cbi(DDRD,PD1);
```

```
    cbi(DDRD,PD2);

    cbi(DDRD,PD3);

    cbi(DDRD,PD6);


volatile unsigned char count=0,count1=0,count2=0,count3=0,count4=0,count5=0;


volatile unsigned char counter1=COUNTER_LOWER_LIMIT;

    volatile unsigned char counter2=COUNTER_LOWER_LIMIT;



TCCR1A |= (1<<COM1A1) | (1<<COM1B1) |(1<<WGM10) | (1<<WGM11) ;


    TCCR1B |= (1<<CS10) | (1<<WGM12);


while(1){


    delay_ms(1000);


 if(counter1 >= COUNTER_UPPER_LIMIT)

  counter1 = COUNTER_LOWER_LIMIT;

 //increase speed by a fixed step

     if(PINC & (1<<PC0) && !(PINC & (1<<PC1))){


       OCR1A = counter1;

       set_FORWARD;

    delay_ms(2000);
```

```
counter1 = counter1+STEP_SIZE;
        count=0;
        count1=0;
        count2=0;
        count3=0;
        count4=0;
        count5=0;

}

if(PINC & (1<<PC1) && !(PINC & (1<<PC0))){

    OCR1B = counter2;
    set_REVERSE;
  delay_ms(2000);
  counter2 = counter2+STEP_SIZE;
        count=0;
        count1=0;
        count2=0;
        count3=0;
        count4=0;
        count5=0;

}
```

```
//stepper1

    if(PIND & (1<<PD0) && !(PIND & (1<<PD1)) && !(PIND & (1<<PD2)) && !(PIND &
(1<<PD3))){

        if(count!=1){


        sbi(OUTPUT_PORT,LED4);

        delay_ms(500);

        cbi(OUTPUT_PORT,LED4);


        sbi(OUTPUT_PORT,LED3);

        delay_ms(500);

        cbi(OUTPUT_PORT,LED3);


        sbi(OUTPUT_PORT,LED2);

        delay_ms(500);

        cbi(OUTPUT_PORT,LED2);


        sbi(OUTPUT_PORT,LED1);

        delay_ms(500);

        cbi(OUTPUT_PORT,LED1);

                        count=count+1;

                count1=0;

                count2=0;

                count3=0;
```

```
            count4=0;
            count5=0;
    }

     }

     // && !(PINC & (1<<PC7))

     if(PIND & (1<<PD1) && !(PIND & (1<<PD0)) && !(PIND & (1<<PD2)) && !(PIND &
(1<<PD3)) && !(PIND & (1<<PD6))){

            if(count1!=2)
            {

    sbi(OUTPUT_PORT,LED4);
    delay_ms(500);
    cbi(OUTPUT_PORT,LED4);

    sbi(OUTPUT_PORT,LED3);
    delay_ms(500);
    cbi(OUTPUT_PORT,LED3);

    sbi(OUTPUT_PORT,LED2);
    delay_ms(500);
    cbi(OUTPUT_PORT,LED2);

    sbi(OUTPUT_PORT,LED1);
```

```
delay_ms(500);
cbi(OUTPUT_PORT,LED1);
            count1=count1+1;
        count=0;
            count2=0;
            count3=0;
            count4=0;
            count5=0;


}


  }


    if(PIND & (1<<PD0) && (PIND & (1<<PD1)) && !(PIND & (1<<PD2)) && !(PIND &
(1<<PD3)) && !(PIND & (1<<PD6))){
    if(count2!=3)
        {

    sbi(OUTPUT_PORT,LED4);
    delay_ms(500);
    cbi(OUTPUT_PORT,LED4);

    sbi(OUTPUT_PORT,LED3);
    delay_ms(500);
    cbi(OUTPUT_PORT,LED3);

    sbi(OUTPUT_PORT,LED2);
```

```
delay_ms(500);

cbi(OUTPUT_PORT,LED2);


sbi(OUTPUT_PORT,LED1);

delay_ms(500);

cbi(OUTPUT_PORT,LED1);

                count2=count2+1;

           count=0;

 count1=0;

                count3=0;

                count4=0;

                count5=0;


}


   }


    if(PIND & (1<<PD2) && !(PIND & (1<<PD1)) && !(PIND & (1<<PD0)) && !(PIND &
(1<<PD3)) && !(PIND & (1<<PD6))){

 if(count3!=4)

          {


 sbi(OUTPUT_PORT,LED4);

 delay_ms(500);

 cbi(OUTPUT_PORT,LED4);


 sbi(OUTPUT_PORT,LED3);
```

```
        delay_ms(500);
        cbi(OUTPUT_PORT,LED3);


        sbi(OUTPUT_PORT,LED2);
        delay_ms(500);
        cbi(OUTPUT_PORT,LED2);


        sbi(OUTPUT_PORT,LED1);
        delay_ms(500);
        cbi(OUTPUT_PORT,LED1);
                count3=count3+1;
                count=0;
                count1=0;
                count2=0;
                count4=0;
                count5=0;
        }


          }


          if(PIND & (1<<PD0) && !(PIND & (1<<PD1)) && (PIND & (1<<PD2)) && !(PIND &
    (1<<PD3)) && !(PIND & (1<<PD6))){
         if(count4!=5)
                 {


         sbi(OUTPUT_PORT,LED4);
         delay_ms(500);
```

```
cbi(OUTPUT_PORT,LED4);


sbi(OUTPUT_PORT,LED3);
delay_ms(500);
cbi(OUTPUT_PORT,LED3);


sbi(OUTPUT_PORT,LED2);
delay_ms(500);
cbi(OUTPUT_PORT,LED2);


sbi(OUTPUT_PORT,LED1);
delay_ms(500);
cbi(OUTPUT_PORT,LED1);
            count4=count4+1;
            count=0;
            count1=0;
            count2=0;
            count3=0;
            count5=0;
}


    }


        if(PIND & (1<<PD0) && (PIND & (1<<PD1)) && (PIND & (1<<PD2)) && (PIND &
(1<<PD3)) && (PIND & (1<<PD6))){
  if(count5!=6)
            {
```

```
sbi(OUTPUT_PORT,LED4);
delay_ms(500);
cbi(OUTPUT_PORT,LED4);

sbi(OUTPUT_PORT,LED3);
delay_ms(500);
cbi(OUTPUT_PORT,LED3);

sbi(OUTPUT_PORT,LED2);
delay_ms(500);
cbi(OUTPUT_PORT,LED2);

sbi(OUTPUT_PORT,LED1);
delay_ms(500);
cbi(OUTPUT_PORT,LED1);
            count5=count5+1;
            count=0;
            count1=0;
            count2=0;
            count3=0;
            count4=0;
}


   }
```

```
//steepper2 forward slow

if(PIND & (1<<PD2) && (PIND & (1<<PD1)) && !(PIND & (1<<PD0)) && !(PIND &
(1<<PD3)) && !(PIND & (1<<PD6))){

{

/* sbi(OUTPUT_PORT1,LED8);

delay_ms(500);

cbi(OUTPUT_PORT1,LED8);

sbi(OUTPUT_PORT1,LED7);

delay_ms(500);

cbi(OUTPUT_PORT1,LED7);

sbi(OUTPUT_PORT1,LED6);

delay_ms(500);

cbi(OUTPUT_PORT1,LED6);

sbi(OUTPUT_PORT1,LED5);

delay_ms(500);

cbi(OUTPUT_PORT1,LED5);

count=0;

count1=0;

count2=0;
```

```
        count3=0;
        count4=0;
        count5=0;*/

        PORTB =0b10000000;
        delay_ms(500);
    PORTB =0b00000000;
        PORTB =0b01000000;
        delay_ms(500);
    PORTB =0b00000000;
        PORTB =0b00100000;
        delay_ms(500);
    PORTB =0b00000000;
        PORTB =0b00010000;
        delay_ms(500);
    PORTB =0b00000000;
        count=0;
        count1=0;
        count2=0;
        count3=0;
        count4=0;
        count5=0;

}

  }
```

```
    //steepper2 forward medium

    if(PIND & (1<<PD0) && (PIND & (1<<PD1)) && (PIND & (1<<PD2)) && !(PIND &
(1<<PD3)) && !(PIND & (1<<PD6))){
  {

    /*sbi(OUTPUT_PORT1,LED8);
    delay_ms(100);
    cbi(OUTPUT_PORT1,LED8);

    sbi(OUTPUT_PORT1,LED7);
    delay_ms(100);
    cbi(OUTPUT_PORT1,LED7);

    sbi(OUTPUT_PORT1,LED6);
    delay_ms(100);
    cbi(OUTPUT_PORT1,LED6);

    sbi(OUTPUT_PORT1,LED5);
    delay_ms(100);
    cbi(OUTPUT_PORT1,LED5);

            count=0;
                count1=0;
```

```
        count2=0;
        count3=0;
        count4=0;
        count5=0;*/

        PORTB =0b10000000;
        delay_ms(100);
    PORTB =0b00000000;
        PORTB =0b01000000;
        delay_ms(100);
    PORTB =0b00000000;
        PORTB =0b00100000;
        delay_ms(100);
    PORTB =0b00000000;
        PORTB =0b00010000;
        delay_ms(100);
    PORTB =0b00000000;
        count=0;
        count1=0;
        count2=0;
        count3=0;
        count4=0;
        count5=0;

}
```

```
}

//steepper2 forward fast

if(PIND & (1<<PD3) && !(PIND & (1<<PD1)) && !(PIND & (1<<PD2)) && !(PIND &
(1<<PD0)) && !(PIND & (1<<PD6))){

{

/* sbi(OUTPUT_PORT1,LED8);

delay_ms(10);

cbi(OUTPUT_PORT1,LED8);

sbi(OUTPUT_PORT1,LED7);

delay_ms(10);

cbi(OUTPUT_PORT1,LED7);

sbi(OUTPUT_PORT1,LED6);

delay_ms(10);

cbi(OUTPUT_PORT1,LED6);

sbi(OUTPUT_PORT1,LED5);

delay_ms(10);

cbi(OUTPUT_PORT1,LED5);

        count=0;

            count1=0;
```

```
count2=0;
count3=0;
count4=0;
count5=0;*/

PORTB =0b10000000;
delay_ms(10);
PORTB =0b00000000;
PORTB =0b01000000;
delay_ms(10);
PORTB =0b00000000;
PORTB =0b00100000;
delay_ms(10);
PORTB =0b00000000;
PORTB =0b00010000;
delay_ms(10);
PORTB =0b00000000;
count=0;
count1=0;
count2=0;
count3=0;
count4=0;
count5=0;

}
```

```
}

//steepper2 reverse slow
    if(PIND & (1<<PD0) && !(PIND & (1<<PD1)) && !(PIND & (1<<PD2)) && (PIND &
(1<<PD3)) && !(PIND & (1<<PD6))){

 {

   /* sbi(OUTPUT_PORT1,LED8);
    delay_ms(500);
    cbi(OUTPUT_PORT1,LED8);

    sbi(OUTPUT_PORT1,LED5);
    delay_ms(500);
    cbi(OUTPUT_PORT1,LED5);

    sbi(OUTPUT_PORT1,LED6);
    delay_ms(500);
    cbi(OUTPUT_PORT1,LED6);

    sbi(OUTPUT_PORT1,LED7);
    delay_ms(500);
    cbi(OUTPUT_PORT1,LED7);*/

                PORTB =0b10000000;
                delay_ms(500);
            PORTB =0b00000000;
```

```
            PORTB =0b00010000;
            delay_ms(500);
        PORTB =0b00000000;
            PORTB =0b00100000;
            delay_ms(500);
        PORTB =0b00000000;
            PORTB =0b01000000;
            delay_ms(500);
        PORTB =0b00000000;
            count=0;
            count1=0;
            count2=0;
            count3=0;
            count4=0;
            count5=0;
  }

    }

        //steepper2 reverse medium
    if(PIND & (1<<PD1) && !(PIND & (1<<PD0)) && !(PIND & (1<<PD2)) && (PIND &
(1<<PD3)) && !(PIND & (1<<PD6))){
 {

  /* sbi(OUTPUT_PORT1,LED8);
   delay_ms(100);
   cbi(OUTPUT_PORT1,LED8);
```

```
sbi(OUTPUT_PORT1,LED5);
delay_ms(100);
cbi(OUTPUT_PORT1,LED5);

sbi(OUTPUT_PORT1,LED6);
delay_ms(100);
cbi(OUTPUT_PORT1,LED6);

sbi(OUTPUT_PORT1,LED7);
delay_ms(100);
cbi(OUTPUT_PORT1,LED7);

        count=0;
            count1=0;
            count2=0;
            count3=0;
            count4=0;
            count5=0;*/

      PORTB =0b10000000;
            delay_ms(100);
        PORTB =0b00000000;
            PORTB =0b00010000;
            delay_ms(100);
        PORTB =0b00000000;
```

```
                    PORTB =0b00100000;

                    delay_ms(100);

                PORTB =0b00000000;

                    PORTB =0b01000000;

                    delay_ms(100);

                PORTB =0b00000000;

                    count=0;

                    count1=0;

                    count2=0;

                    count3=0;

                    count4=0;

                    count5=0;


    }


      }


                      //steepper2 reverse fast

      if(PIND & (1<<PD0) && (PIND & (1<<PD1)) && !(PIND & (1<<PD2)) && (PIND &
(1<<PD3)) && !(PIND & (1<<PD6))){

   {


     /* sbi(OUTPUT_PORT1,LED8);

      delay_ms(10);

      cbi(OUTPUT_PORT1,LED8);


      sbi(OUTPUT_PORT1,LED5);
```

```
delay_ms(10);
cbi(OUTPUT_PORT1,LED5);

sbi(OUTPUT_PORT1,LED6);
delay_ms(10);
cbi(OUTPUT_PORT1,LED6);

sbi(OUTPUT_PORT1,LED7);
delay_ms(10);
cbi(OUTPUT_PORT1,LED7);

        count=0;
            count1=0;
            count2=0;
            count3=0;
            count4=0;
            count5=0;*/

        PORTB =0b10000000;
            delay_ms(10);
          PORTB =0b00000000;
            PORTB =0b00010000;
            delay_ms(10);
          PORTB =0b00000000;
            PORTB =0b00100000;
            delay_ms(10);
```

```
PORTB =0b00000000;
        PORTB =0b01000000;
        delay_ms(10);
PORTB =0b00000000;
        count=0;
        count1=0;
        count2=0;
        count3=0;
        count4=0;
        count5=0;


}


    }


//flashlight
 if(PIND & (1<<PD3) && !(PIND & (1<<PD1)) && (PIND & (1<<PD2)) && !(PIND & (1<<PD0))
&& !(PIND & (1<<PD6))){
    sbi(PORTA,PA0);
            cbi(PORTA,PA1);
            cbi(PORTA,PA2);
            cbi(PORTA,PA3);
            cbi(PORTA,PA4);
            cbi(PORTA,PA5);
            cbi(PORTA,PA6);
            cbi(PORTA,PA7);
```

```
}
if(PIND & (1<<PD0) && !(PIND & (1<<PD1)) && (PIND & (1<<PD2)) && (PIND & (1<<PD3))
&& !(PIND & (1<<PD6))){
    sbi(PORTA,PA1);
        sbi(PORTA,PA0);
            cbi(PORTA,PA2);
        cbi(PORTA,PA3);
        cbi(PORTA,PA4);
        cbi(PORTA,PA5);
        cbi(PORTA,PA6);
        cbi(PORTA,PA7);
        }
if(PIND & (1<<PD1) && !(PIND & (1<<PD0)) && (PIND & (1<<PD2)) && (PIND & (1<<PD3))
&& !(PIND & (1<<PD6))){
    sbi(PORTA,PA1);
        sbi(PORTA,PA0);

        sbi(PORTA,PA2);
        cbi(PORTA,PA3);
        cbi(PORTA,PA4);
        cbi(PORTA,PA5);
        cbi(PORTA,PA6);
        cbi(PORTA,PA7);
        }

if(PIND & (1<<PD0) && (PIND & (1<<PD1)) && (PIND & (1<<PD2)) && (PIND & (1<<PD3))
&& !(PIND & (1<<PD6))){
    sbi(PORTA,PA1);
```

```
            sbi(PORTA,PA0);

            sbi(PORTA,PA2);

            sbi(PORTA,PA3);

                    cbi(PORTA,PA4);

            cbi(PORTA,PA5);

            cbi(PORTA,PA6);

            cbi(PORTA,PA7);

            }


    if(PIND & (1<<PD6) && !(PIND & (1<<PD1)) && !(PIND & (1<<PD2)) && !(PIND & (1<<PD3))
&& !(PIND & (1<<PD0))){

      sbi(PORTA,PA1);

            sbi(PORTA,PA0);

            sbi(PORTA,PA2);

            sbi(PORTA,PA3);

             sbi(PORTA,PA4);

                    cbi(PORTA,PA5);

            cbi(PORTA,PA6);

            cbi(PORTA,PA7);

    }

    if(PIND & (1<<PD0) && !(PIND & (1<<PD1)) && !(PIND & (1<<PD2)) && !(PIND & (1<<PD3))
&& (PIND & (1<<PD6))){

     sbi(PORTA,PA1);

                 sbi(PORTA,PA1);

            sbi(PORTA,PA0);

            sbi(PORTA,PA2);

            sbi(PORTA,PA3);
```

```
sbi(PORTA,PA4);
sbi(PORTA,PA5);
cbi(PORTA,PA6);
cbi(PORTA,PA7);
}
if(PIND & (1<<PD1) && !(PIND & (1<<PD0)) && !(PIND & (1<<PD2)) && !(PIND & (1<<PD3))
&& (PIND & (1<<PD6))){
sbi(PORTA,PA6);
sbi(PORTA,PA1);
sbi(PORTA,PA0);
sbi(PORTA,PA2);
sbi(PORTA,PA3);
sbi(PORTA,PA4);
sbi(PORTA,PA5);
sbi(PORTA,PA6);
cbi(PORTA,PA7);
}
if(PIND & (1<<PD0) && (PIND & (1<<PD1)) && !(PIND & (1<<PD2)) && !(PIND & (1<<PD3))
&& (PIND & (1<<PD6))){
sbi(PORTA,PA6);
sbi(PORTA,PA1);
sbi(PORTA,PA0);
sbi(PORTA,PA2);
sbi(PORTA,PA3);
sbi(PORTA,PA4);
sbi(PORTA,PA5);
sbi(PORTA,PA6);
```

```
            sbi(PORTA,PA7);
      }
  }
      return 0;
}
```