Programming for Automatic over speed Control System for Safety in Automobiles

[1]Pushkar P. Bhatt, [2]Kunjan A. Chaudhari, [3]Kaushal A. Chaudhary, [4]Siddharth S. Goyal, [5]Akash B. Pandey

Department of Mechanical Engineering, Faculty of Technology & Engineering, M. S. University, Baroda, Gujarat
Email : [1]pushkarbhatt07@gmail.com, [2]kunjan.chaudhari@gmail.com, [3]chaudharykaushal20@gmail.com, [4]siddgoyal@gmail.com, [5]akashpandey@gmail.com

Abstract-The introduction of Anti-lock Braking System(ABS) has provided building blocks for a wide variety of braking control systems. Additional hardware allows brake pressure to be increased or reduced as per demand. Additional software control algorithms and sensors allow traction control (TC),electronic brake force distribution (EBD) ,brake assist (BA) and electronic stability control (ESC) functions to be added.

Automatic Over speed Control System (A.O.C.S.) is a concept that stresses on preparing a control system that is based on automatic braking for a motor-cycle. Here automation is done by a Microcontroller. Whenever the speed of a vehicle increases beyond a pre-defined critical high speed, the Microcontroller based system actuates the clutch as well as brake and brings speed of an automobile down to a lower pre-defined safe speed. As applied to a 4-stroke, manual transmission motor-cycle, the clutch and brake would be actuated by means of stepper motors, which would be controlled by a micro-controller based circuit, programmed to read the signals from an infrared based speed sensor.

KEYWORDS: Automatic Over speed Control System (A.O.C.S.), Microcontroller, Anti-lock Braking System (A.B.S.), traction control (TC), electronic brake force distribution (EBD), brake assist (BA) and electronic stability control (ESC).

I. INTRODUCTION

Safety of the vehicle is of prime importance. Safety basically focuses on controlling speed and effective braking systems. With increase in amount of speed there also increases the chances of occurrence of accidents. As per the saying, “Speed thrills but kills”, over speed of the vehicle can lead to very severe accidents. There is more number of accidents that are caused by uncontrollable speeding than due to other reasons [1.]. Severe accidents are caused due to this which can be life threatening as well. Prime reasons behind such accidents are either by negligence of the driver, or by failure of braking system to stop the vehicle in time. Here comes the role of safety of the vehicle which reduces severe mishaps. Nowadays manufacturers of Cars as well as Motor cycles also count on such technologies and equipments which assures safety at high speed. The use of proper braking system and by controlled speed reduction techniques is the key to solve such problems. The use of Anti-lock braking system in cars is the current trend to increase safety of the car. There are also other solutions like Automatic Braking Systems which can be useful as well as helpful [3].

Automatic braking technologies combine sensors and brake controls to help prevent high speed collisions. Some automatic braking systems can prevent collisions altogether, but most of them are designed to simply reduce the speed of a vehicle before it hits something. Since high speed crashes are more likely to be fatal than low speed collisions, automatic braking systems can save lives and reduce the amount of property damage that occurs during an accident. Some of these systems provide braking assistance to the driver, and others are actually capable of activating the brakes with no driver input. Each car manufacturer has its own automatic braking system technology, but they all rely on some type of sensor input [4.]. Some of these systems use lasers, others use radar, and some even use video data. This sensor input is then used to determine if there are any objects present in the path of the vehicle. If an object is detected, the system can then determine if the speed of the vehicle is greater than the speed of the object in front of it. A significant speed differential may indicate that a collision is likely to occur, in which case the system is capable of automatically activating the brakes [2].

II. CONCEPT AND METHODOLOGY

The safety in automobiles can also be ensured by introducing A.O.C.S. which stands for Automatic Over speed Control System on which this system is based. It basically controls the speed of the vehicle by continuously checking it through sensors which sends the signal to the input of Microcontroller. The Microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Whenever the speed is increased above the predefined critical speed, the Microcontroller through its logic circuits sends the signals to the actuators via its output pins. The actuators apply necessary force on the brakes and reduce the speed to its safe value. This concept consists of three major systems: - Sensor, Microcontroller and Actuators (Stepper Motor) [6].

The sensor used in this system is optical type infrared sensor .It consists of an emitter and receiver. The circuit is completed whenever the signal sent by the emitter is struck by an object and is reflected back to the receiver. The receiver sends the signal to the Microcontroller for further Processing [8].

The Arduino Uno is a microcontroller board based on the ATmega328.Arduino has an 8-bit architecture .It acts as brain to the system such as CPU of the Computer and controls all the actions of the system such as sensing and actuating [5].

The Actuators here are Stepper motors in which output of Microcontroller is connected. A stepper motor is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any feedback sensor, as long as the motor is carefully sized to the application [7].

In our project the whole system is mounted on a four-stroke motor cycle named “Kawasaki Bajaj 4-S”. By using automatic over speed control system, effective controlling by automation can be easily achieved. Moreover, the automatic braking system not only reduces human effort but also helps in proper application of the brake at the required time and with proper amount of force [10.].

III. SYSTEM OPERATION

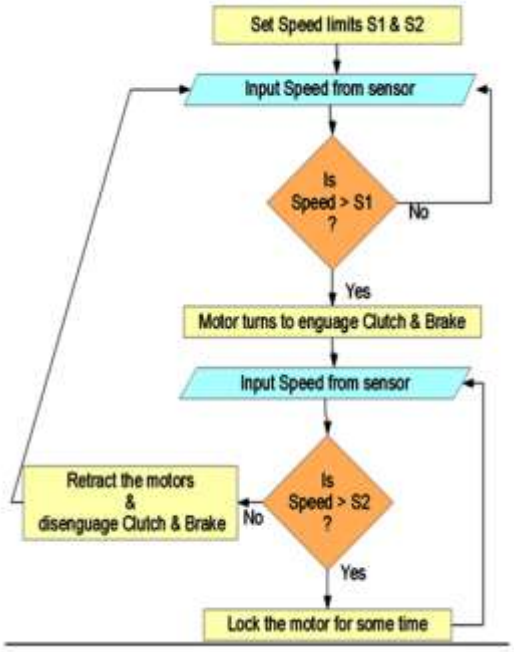

Fig. 1. Control Logic Flowchart

The above flowchart indicates the control logic used in the system.

When 5V is applied to the sensor system, the IR LED will get reverse biased and will start to emit IR radiations. The incident radiation gets reflected by a reflecting surface mounted on the rear wheel, and is received by a IR photodiode. Because of this, the collector emitter circuit gets completed, and current is conducted with a transfer ratio of about 0.1.

It generates a wave which is shown in fig.3.

This wave is input to the Arduino Microcontroller via an analog pin.

It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

Then, the equivalent digital value is used as data for the controller to calculate the rpm.

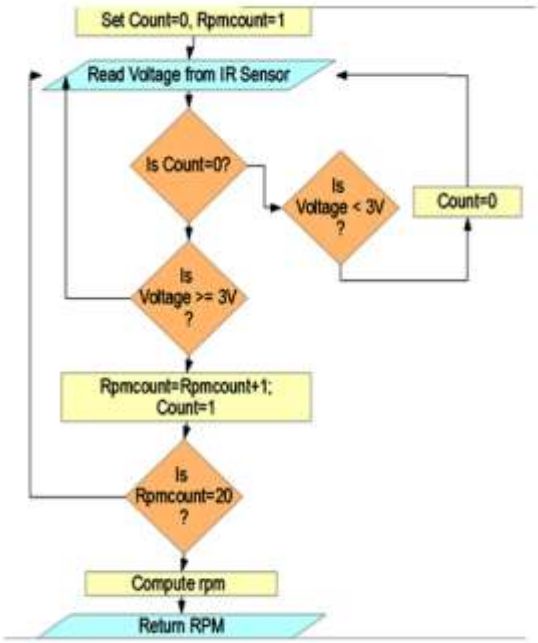

Fig.2. RPM Calculation

When the rpm of the wheel increases beyond the set limit, the controller sends a signal to the stepper motors via the driver card, and turns it by a certain number of predefined steps at a pre defined speed thereby engaging the clutch and brake.

There are two stepper motors, one of 10 Kg-cm torque, connected to the clutch, other of 2Kg-cm torque connected to the brake lever.

Both 6-wire motors are running in unipolar mode, driven by a single driven by a single driver card. Both centre taps are maintained at 12 V potential difference by the motorcycle's battery.

To drive these motors, the microcontroller sends a sequential digital signal through its digital ports (here pins 8,9,10,11 respectively) which are further connected to the stepper motor input wires in the sequence of Blue, Green, Red, Orange. During each signal by the microcontroller, the stepper motor moves by 1.8 ^{o} or 1 step.

After the motors turn by the requisite number of turns they get locked by the controller. They remain locked while the microcontroller again calculates the reduced rpm, and till it reaches a value below the set limit. Then, the motors retract, thereby disengaging the clutch and brake.

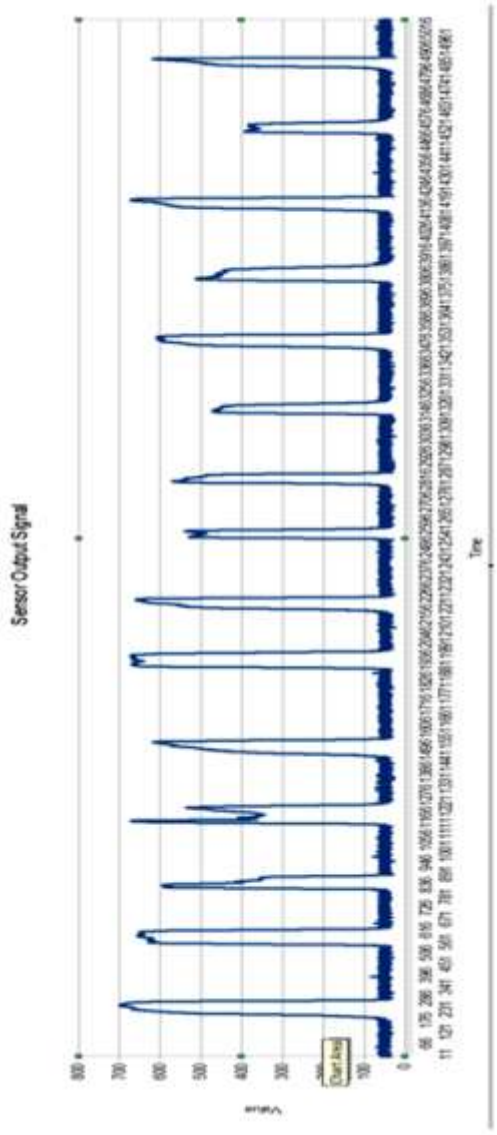

Fig. 3. Signal Wave

IV. PROGRAMMING

setup()

The setup() function is called when a sketch starts. It is used to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each powerup or reset of the Arduino board.

Example

```
int buttonPin = 3;

void setup()
{
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}
```

```
void loop()
{
  // ...
}
```

LOOP()

After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing the program to change and respond. It is used to actively control the Arduino board.

Example

```
 const int buttonPin = 3;

// setup initializes serial and the button pin
void setup()
{
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

// loop checks the button pin each time,
// and will send serial if it is pressed
void loop()
{
  if (digitalRead(buttonPin) == HIGH)
    Serial.write('H');
  else
    Serial.write('L');

  delay(1000);
}
```

do - while

The do loop works in the same manner as the while loop, with the exception that the condition is tested at the end of the loop, so the do loop will always run at least once.

```
do
{
   // statement block
} while (test condition);
```

Example

```
do
{
  delay(50);          // wait for sensors to stabilize
  x = readSensors();  // check the sensors

} while (x < 100);
```

Return

It terminates a function and returns a value from a function to the calling function, if desired.

Syntax:

```
return;
return value; // both forms are valid
```

Parameters

value: any variable or constant type

Examples:

A function to compare a sensor input to a threshold

```
 int checkSensor(){
    if (analogRead(0) > 400) {
       return 1;
    else{
       return 0;
    }
}

void loop(){
// brilliant code idea to test here

return;
// the rest of a dysfunctional sketch here
// this code will never be executed
}
```

#include

#include is used to include outside libraries in your sketch. This gives the programmer access to a large group of standard C libraries (groups of pre-made functions), and also libraries written especially for Arduino.

Note that #include has no semicolon terminator, and the compiler will yield cryptic error messages if you add one.

analogRead()

Description

It reads the value from the specified analog pin. The Arduino board contains a 6 channel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().

It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

Syntax

analogRead(pin)

Parameters

pin: the number of the analog input pin to read from (0 to 5 on most boards, 0 to 7 on the Mini and Nano, 0 to 15 on the Mega)

Returns

int (0 to 1023)

Note

If the analog input pin is not connected to anything, the

value returned by analogRead() will fluctuate based on a number of factors (e.g. the values of the other analog inputs, how close your hand is to the board, etc.).

Example

```
 int analogPin = 3;     // potentiometer wiper (middle terminal) connected to analog pin 3
                        // outside leads to ground and +5V
int val = 0;           // variable to store the value read

void setup()
{
  Serial.begin(9600);          //  setup serial
}

void loop()
{
  val = analogRead(analogPin);    // read the input pin
  Serial.println(val);            // debug value
}
```

millis()

Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

Returns:

Number of milliseconds since the program started (unsigned long).

Example:

```
unsigned long time;

void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Time: ");
  time = millis();
  //prints time since program started
  Serial.println(time);
  // wait a second so as not to send massive amounts of data
  delay(1000);
}
```

delay()

Pauses the program for the amount of time (in miliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

Syntax:

delay(ms)

Parameters:

ms: the number of milliseconds to pause (unsigned long)

Example:

```
int ledPin = 13;                // LED connected to digital pin 13

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Program Used in the System

```
// Include the Stepper Library
#include<Stepper.h>
//Setting the speed limits. Upper limit is lim1, Lower limit is Lim 2
int lim1=120,lim2=100;
float rpmvalue;
//Define the parameters of stepper motor called stepper1
Stepper stepper1(200,8,9,10,11);
int delayvalue=300;
int stepperspeed=10;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  //define the speed of the stepper
  stepper1.setSpeed(stepperspeed);
  //read rpm value
  rpmvalue=rpm(20);
  Serial.println("Entry RPM");
  Serial.println(rpmvalue);

  if(rpmvalue>lim1)
  {
    //Turn the stepper to enguage clutch and brake
    stepper1.step(200);
    do
    {
      //Lock the motor for some time
      delay(delayvalue);
      rpmvalue=rpm(10);
      Serial.println("Delay Rpm");
      Serial.println(rpmvalue);
    }
    while ( rpmvalue>lim2);
  //Disenguage the clutch and brake
   stepper1.step(-200);
  }
}
```

```
//Subprogram to calculate the RPM
unsigned long int rpm(byte i)
{

 byte rpmcount,count=0;
 unsigned long timeold,time;
 float rpmval;
 int sensorval;

 rpmcount = 1;

 timeold = millis();

 while(rpmcount<=i)
  {
   delay(100);
   sensorval=analogRead(1);
   if(count==0)
     {
      if(sensorval>=800)
       {
        rpmcount=rpmcount+1;
        count=1;
       }
     }

   if(sensorval<800)
     {
      count=0;
     }
  }

 time=millis()-timeold;
 rpmval = (60000/time)*rpmcount;

 return rpmval;
}
```

V. CONCLUSION

By using this programming in Arduino Microcontroller, We can generate system of Speed Control and automatic Braking system with the help of Stepper motor and servo motors.

Speed control and automatic braking systems for two wheelers will become a common scenario in the market place. With the increasing number of accidents due to over speeding and also over speed thefts, this system has a bright future, going by the saying “prevention is better than cure”.

VI. ACKNOWLEDGEMENT

We would like to take this opportunity to bestow our acknowledgement to all the people who directly or indirectly availed us in making our project feasible on the above subject.

The assiduous help presumed Shri G. D. KARHADKAR, Head of the Department, Mech. Engg Deptt., Faculty of Tech. & Engg., M.S. University, Vadodara for being the inevitable guide of successful consummation of our project.

REFERENCES

[1] C. Grover ,I. Knight, F Okora,I. Simmons, G. Couper, P. Massie., Automatic Emergency Braking System: Technical Requirements, Costs and Benefits. Published project Rep ort, PPR 227 Contract ENTR/05/17.01 ,TRL limited. Clients: DG Enterprise.,2008.

[2] T. C. Goldsmith., Automated Vehicle Guidance (AVCS) - The Real Automobile. http://www.azinet.com, 1998

[3] M. Habibullah Pagarkar, Kaushal Parekh, Jogen Shah, Jignasa Desai, Prarthna Advani, Siddhesh Sarvankar, Nikhil Ghate, Automated Vehicle Control System http://www.cs.jhu.edu

[4] Muhammad Ali Mazidi, Janice Gillespie Mazidi, and Rolin D. Mckinlay, The 8051 Micro controller & Embedded System Using Assembly and C, Pearson Education, 2nd Edition, 2009.

[5] http://www.arduino.cc/

[6] http://www.engineersgarage.com

[7] http://www.lagacycncwoodworking.com

[8] http://www.robotoid.com

[9] http://www.nmbtc.com/nmb componentapplications/medical/motors-laboratory-automationsystems

[10] http://www.solarbotics.net/library/ pdflib/pdf/motorbas.pdf

[11] http://www.stepperworld.com/ Tutorials/pgUnipolarTutorial.html

[12] http://www.embedded.com/design/ embedded/4024586/Exploring-optical-and magnetic-sensors.