write code in arduino to meseuare motorcycle speed and distance using hall effect sensor

Sure, here is some example code that you can use to measure motorcycle speed and distance using a Hall Effect sensor with Arduino:

```c++
const int hallPin = 2; // Hall Effect sensor connected to digital pin 2
float wheelCircumference = 2.07; // Wheel circumference in meters (modify to match
volatile int hallCount = 0; // Hall Effect sensor interrupt counter
float distance = 0; // Distance traveled in meters
unsigned long lastTime = 0; // Time of last speed measurement
float speed = 0; // Motorcycle speed in meters per second

void setup() {
  pinMode(hallPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(hallPin), countHall, RISING);
  Serial.begin(9600);
}

void loop() {
  unsigned long now = millis();
  if (now - lastTime >= 1000) { // Calculate speed every second
    detachInterrupt(digitalPinToInterrupt(hallPin)); // Disable interrupt during ca
    speed = (float)hallCount * wheelCircumference / (float)1000; // Calculate speed
    hallCount = 0; // Reset interrupt counter
    lastTime = now; // Update time of last speed measurement
    attachInterrupt(digitalPinToInterrupt(hallPin), countHall, RISING); // Re-enabl
  }
  distance = (float)hallCount * wheelCircumference / (float)1000; // Calculate dist
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.print(" m, Speed: ");
  Serial.print(speed);
  Serial.println(" m/s");
  delay(100); // Adjust delay as needed
}

void countHall() {
  hallCount++; // Increment interrupt counter
}
```

In this code, we define the digital pin that the Hall Effect sensor is connected to (in this example, pin 2), the circumference of the motorcycle's wheel (in meters), and a few variables for storing the distance traveled and motorcycle speed.

In the `setup()` function, we set the pin mode for the Hall Effect sensor and attach an interrupt to it using the `attachInterrupt()` function. The interrupt is triggered on the rising edge of the sensor signal and calls the `countHall()` function.

In the `loop()` function, we calculate the motorcycle speed once per second using the interrupt counter and the wheel circumference. We also calculate the distance traveled by the motorcycle using the same variables. Finally, we print the distance and speed values to the serial monitor.

The `countHall()` function simply increments the interrupt counter every time the interrupt is triggered.

Note that you may need to adjust the values of the `wheelCircumference` variable and the delay in the `loop()` function to match your motorcycle's wheel size and the desired update rate for the speed and distance measurements.

write the above code for atmgea32 in c

Sure! Here is an example code in C for measuring motorcycle speed and distance using a Hall Effect sensor with the ATMega32 microcontroller:

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

volatile int hallCount = 0; // Hall Effect sensor interrupt counter
float wheelCircumference = 2.07; // Wheel circumference in meters (modify to match
float distance = 0; // Distance traveled in meters
unsigned long lastTime = 0; // Time of last speed measurement
float speed = 0; // Motorcycle speed in meters per second

int main(void) {
  DDRD &= ~(1 << PD2); // Set PD2 as input pin
  PORTD |= (1 << PD2); // Enable pull-up resistor on PD2
  TCCR1B |= (1 << ICES1); // Set input capture edge to rising edge
  TIMSK |= (1 << TICIE1); // Enable input capture interrupt
  sei(); // Enable global interrupts
  TCCR1B |= (1 << CS11) | (1 << CS10); // Start timer with prescaler of 64
  while (1) {
    unsigned long now = TCNT1;
    if (now - lastTime >= 15625) { // Calculate speed every second
      cli(); // Disable interrupts during calculation
      speed = (float)hallCount * wheelCircumference / (float)1000; // Calculate spe
      hallCount = 0; // Reset interrupt counter
      lastTime = now; // Update time of last speed measurement
      sei(); // Enable interrupts
    }
    distance = (float)hallCount * wheelCircumference / (float)1000; // Calculate di
    // Output distance and speed values here
  }
  return 0;
}

ISR(TIMER1_CAPT_vect) {
  hallCount++; // Increment interrupt counter
}
```

In this code, we define the input pin connected to the Hall Effect sensor (in this example, pin PD2), the circumference of the motorcycle's wheel (in meters), and a few variables for storing the distance traveled and motorcycle speed.

In the `main()` function, we configure the input pin as an input and enable the pull-up resistor. We also configure the timer to trigger an interrupt on the rising edge of the Hall Effect sensor signal using the input capture feature. The interrupt vector is defined using the `ISR()` macro and simply increments the interrupt counter.

In the main loop, we calculate the motorcycle speed once per second by reading the timer value and checking if one second has elapsed. We disable interrupts during this calculation to ensure that the interrupt counter is not changed while we are calculating the speed. After calculating the speed, we reset the interrupt counter and update the time of the last speed measurement. Finally, we enable interrupts again. We also calculate the distance traveled by the motorcycle using the same variables.

Note that you may need to adjust the values of the `wheelCircumference` variable and the time delay in the `if` statement to match your motorcycle's wheel size and the desired update rate for the speed and distance measurements. You will also need to output the distance and speed values using your desired output method (e.g. serial output or LCD display).